Deutsch

Fujitsu Software BS2000

openUTM V7.0

Einsatz von UTM-Anwendungen auf BS2000-Systemen

Benutzerhandbuch

Stand der Beschreibung:
openUTM V7.0A00

Ausgabe November 2019

Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an bs2000.info@fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2015

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001: 2015 erfüllt.

Copyright und Handelsmarken

Inhaltsverzeichnis

Einsatz von UTM-Anwendungen auf BS2000-Systemen

1 Einleitung

Die IT-Infrastruktur heutiger Unternehmen als Herzstück und Motor des Geschäftes muss den Anforderungen des digitalen Zeitalters gerecht werden. Dabei muss sie mit vermehrten Datenmengen genauso zurechtkommen wie mit verschärften Anforderungen aus dem Umfeld, z.B. Einhaltung von Compliance-Vorgaben. Ebenso muss die Möglichkeit der kurzfristigen Integration weiterer Applikationen gegeben sein. Und alles dies unter dem Gesichtspunkt einer gewährleisteten Sicherheit.
Somit bestehen wesentliche Anforderungen an eine moderne IT-Infrastruktur u.a. aus

- Flexibilität und schier grenzenloser Skalierbarkeit auch für zukünftige Anforderungen
- hohe Robustheit bei höchster Verfügbarkeit
- absoluter Sicherheit in allen Belangen
- Anpassbarkeit an individuelle Bedürfnisse
- Verursachen geringer Kosten

Fujitsu bietet zur Bewältigung dieser Herausforderungen ein umfangreiches Portfolio innovativer Enterprise Hardware, Software und Support Services im Umfeld unserer Enterprise Mainframe Plattformen an und ist damit Ihr

- verlässlicher Service Provider, der Sie langfristig, flexibel und innovativ beim Betrieb der Mainframe-basierten Kernanwendungen Ihres Geschäftes unterstützt,
- optimaler Partner für die gemeinsame Abdeckung der Anforderungen einer Digitalen Transformation und
- langfristiger Partner aufgrund kontinuierlicher Anpassung moderner Schnittstellen, die eine moderne IT Landschaft mit all ihren Anforderungen mit sich bringt.

Mit openUTM stellt Ihnen Fujitsu eine vielfach erprobte und bewährte Lösung aus dem Middleware-Bereich zur Verfügung.

Die High-End-Plattform für Transaktionsverarbeitung openUTM bietet eine Ablaufumgebung, die all diesen Anforderungen moderner unternehmenskritischer Anwendungen gewachsen ist, denn openUTM verbindet alle Standards und Vorteile von transaktionsorientierten Middleware-Plattformen und Message Queuing Systemen:

- Konsistenz der Daten und der Verarbeitung
- Hohe Verfügbarkeit der Anwendungen
- Hohen Durchsatz auch bei großen Benutzerzahlen, d.h. höchste Skalierbarkeit
- Flexibilität bezüglich Änderungen und Anpassungen des IT-Systems

Eine UTM-Anwendung auf Unix-, Linux- und Windows-Systemen kann auf einem einzelnen Rechner als stand-alone UTM-Anwendung oder auf mehreren Rechnern gleichzeitig als UTM-Cluster-Anwendung betrieben werden.

openUTM ist Teil des umfassenden Angebots von **openSEAS**. Gemeinsam mit der Oracle Fusion Middleware bietet openSEAS die komplette Funktionalität für Anwendungsinnovation und moderne Anwendungsentwicklung. Im Rahmen des Produktangebots **openSEAS** nutzen innovative Produkte die ausgereifte Technologie von openUTM:

- BeanConnect ist ein Adapter gemäß der Java EE Connector Architecture (JCA) und bietet den standardisierten Anschluss von UTM-Anwendungen an Java EE Application Server. Dadurch können bewährte Legacy-Anwendungen in neue Geschäftsprozesse integriert werden.

- Bestehende UTM-Anwendungen können unverändert ins Web übernommen werden. Mit dem UTM-HTTP Interface und dem Produkt WebTransactions stehen in openSEAS zwei Alternativen zur Verfügung, welche es ermöglichen, bewährte Host-Anwendungen flexibel in neuen Geschäftsprozessen und modernen Einsatzszenarien zu nutzen.

Die Produkte BeanConnect und WebTransactions werden im Leistungsüberblick kurz dargestellt. Für diese Produkte gibt es eigene Handbücher.

Wenn im Folgenden von Linux-System bzw. Linux-Plattform die Rede ist, dann ist darunter eine Linux-Distribution wie z.B. SUSE oder Red Hat zu verstehen.

Wenn im Folgenden von Windows-System bzw. Windows-Plattform die Rede ist, dann sind damit alle Windows-Varianten gemeint, auf denen openUTM zum Ablauf kommt.

Wenn im Folgenden von Unix-System bzw. Unix-Plattform die Rede ist, dann ist darunter ein Unix-basiertes Betriebssystem wie z.B. Solaris oder HP-UX zu verstehen.

1.1 Zielgruppe und Konzept des Handbuchs

Dieses Handbuch richtet sich an Anwendungsplaner, Anwendungsentwickler, Anwender und Betreuer von UTM-Anwendungen.

Es enthält alle Informationen, um ein UTM-Anwendungsprogramm auf einem BS2000-System zu erzeugen und eine UTM-Anwendung einzusetzen.

Dieses Handbuch gibt Ihnen in den ersten Kapiteln einen Überblick darüber, wie Sie eine UTM-Anwendung strukturieren und binden und welche Dateien zum Betrieb einer Anwendung notwendig sind. Jeweils eigene Kapitel befassen sich mit dem Starten und Beenden einer UTM-Anwendung und mit dem Programmaustausch bei laufender Anwendung. Die Besonderheiten, die Sie beim Betrieb einer UTM-Datenbank-Anwendung beachten müssen, sind zentral in einem gleichlautenden Kapitel zusammengestellt.

Ausführlich wird darauf eingegangen, wie sich Terminal-Benutzer und andere Clients an eine UTM-Anwendung anmelden können, wie Sie Ihre Anwendung im laufenden Betrieb aktualisieren können und wie sich Ihre Anwendung in Fehlersituationen verhält.

Zusätzlich gibt es eigene Kapitel über die Tools, die Ihnen für den Betrieb und die Kontrolle einer UTM-Anwendung zur Verfügung stehen.

Kenntnisse des Betriebssystems werden vorausgesetzt.

1.2 Wegweiser durch die Dokumentation zu openUTM

In diesem Abschnitt erhalten Sie einen Überblick über die Handbücher zu openUTM und zum Produktumfeld von openUTM.

1.2.1 openUTM-Dokumentation

Die openUTM-Dokumentation besteht aus Handbüchern, den Online-Hilfen für den grafischen Administrationsarbeitsplatz openUTM WinAdmin und das grafische Administrationstool WebAdmin sowie Freigabemitteilungen.

Es gibt Handbücher und Freigabemitteilungen, die für alle Plattformen gültig sind, sowie Handbücher und Freigabemitteilungen, die jeweils für BS2000-Systeme bzw. für Unix-, Linux- und Windows-Systeme gelten.

Sämtliche Handbücher sind im Internet verfügbar unter der Adresse https://bs2manuals.ts.fujitsu.com. Für die Plattform BS2000 finden Sie die Handbücher auch auf der Softbook-DVD.

Die folgenden Abschnitte geben einen Aufgaben-bezogenen Überblick über die Dokumentation zu openUTM V7.0.

Eine vollständige Liste der Dokumentation zu openUTM finden Sie im Literaturverzeichnis.

Einführung und Überblick

Das Handbuch **Konzepte und Funktionen** gibt einen zusammenhängenden Überblick über die wesentlichen Funktionen, Leistungen und Einsatzmöglichkeiten von openUTM. Es enthält alle Informationen, die Sie zum Planen des UTM-Einsatzes und zum Design einer UTM-Anwendung benötigen. Sie erfahren, was openUTM ist, wie man mit openUTM arbeitet und wie openUTM in die BS2000-, Unix-, Linux- und Windows-Plattformen eingebettet ist.

Programmieren

- Zum Erstellen von Server-Anwendungen über die KDCS-Schnittstelle benötigen Sie das Handbuch **Anwendungen programmieren mit KDCS für COBOL, C und C++**, in dem die KDCS-Schnittstelle in der für COBOL, C und C++ gültigen Form und die Programmschnittstelle UTM-HTTP beschrieben sind. Die KDCS-Schnittstelle umfasst sowohl die Basisfunktionen des universellen Transaktionsmonitors als auch die Aufrufe für verteilte Verarbeitung. Es wird auch die Zusammenarbeit mit Datenbanken beschrieben.
  Die Programm-Schnittstelle UTM-HTTP stellt Funktionen zur Verfügung, die für die Kommunikation mit HTTP-Clients verwendet werden können.
- Wollen Sie die X/Open-Schnittstellen nutzen, benötigen Sie das Handbuch **Anwendungen erstellen mit X/Open-Schnittstellen**. Es enthält die openUTM-spezifischen Ergänzungen zu den X/Open-Programmschnittstellen TX, CPI-C und XATMI sowie Hinweise zu Konfiguration und Betrieb von UTM-Anwendungen, die X/Open-Schnittstellen nutzen. Ergänzend dazu benötigen Sie die X/Open-CAE-Spezifikation für die jeweilige X/Open-Schnittstelle.
- Wenn Sie Daten auf Basis von XML austauschen wollen, benötigen Sie das Dokument **XML für openUTM**. Darin werden die C- und COBOL-Aufrufe beschrieben, die zum Bearbeiten von XML-Dokumenten benötigt werden.
- Für BS2000-Systeme gibt es Ergänzungsbände für die Programmiersprachen Assembler, Fortran, Pascal-XT und PL/1.

Konfigurieren

Zur Definition von Konfigurationen steht Ihnen das Handbuch **Anwendungen generieren** zur Verfügung. Darin ist beschrieben, wie Sie mit Hilfe des UTM-Tools KDCDEF sowohl für eine stand-alone UTM-Anwendung als auch für eine UTM-Cluster-Anwendung auf Unix-, Linux- und Windows-Systemen.

- die Konfiguration definieren,
- die KDCFILE erzeugen,
- und im Falle einer UTM-Cluster-Anwendung die UTM-Cluster-Dateien erzeugen.

Zusätzlich wird gezeigt, wie Sie wichtige Verwaltungs- und Benutzerdaten mit Hilfe des Tools KDCUPD in eine neue KDCFILE übertragen, z.B. beim Umstieg auf eine neue Version von openUTM oder nach Änderungen in der Konfiguration. Für eine UTM-Cluster-Anwendung wird außerdem gezeigt, wie Sie diese Daten mit Hilfe des Tools KDCUPD in die neuen UTM-Cluster-Dateien übertragen.

Binden, Starten und Einsetzen

Um UTM-Anwendungen einsetzen zu können, benötigen Sie für das betreffende Betriebssystem (BS2000- bzw. Unix-, Linux- oder Windows-Systeme) das Handbuch **Einsatz von UTM-Anwendungen**.
Dort ist beschrieben, wie man ein UTM-Anwendungsprogramm bindet und startet, wie man sich bei einer UTM-Anwendung an- und abmeldet und wie man Anwendungsprogramme strukturiert und im laufenden Betrieb austauscht. Außerdem enthält es die UTM-Kommandos, die dem Terminal-Benutzer zur Verfügung stehen. Zudem wird ausführlich auf die Punkte eingegangen, die beim Betrieb von UTM-Cluster-Anwendungen zu beachten sind.

Administrieren und Konfiguration dynamisch ändern

- Für das Administrieren von Anwendungen finden Sie die Beschreibung der Programmschnittstelle zur Administration und die UTM-Administrationskommandos im Handbuch **Anwendungen administrieren**. Es informiert über die Erstellung eigener Administrationsprogramme für den Betrieb einer stand-alone UTM-Anwendung oder einer UTM-Cluster-Anwendung sowie über die Möglichkeiten, mehrere UTM-Anwendungen zentral zu administrieren. Darüber hinaus beschreibt es, wie Sie Message Queues und Drucker mit Hilfe der KDCS-Aufrufe DADM und PADM administrieren können.
- Wenn Sie den grafischen Administrationsarbeitsplatz **openUTM WinAdmin** oder die funktional vergleichbare Web-Anwendung **openUTM WebAdmin** einsetzen, dann steht Ihnen folgende Dokumentation zur Verfügung:
  - Die **WinAdmin-Beschreibung** und die **WebAdmin-Beschreibung** bieten einen umfassenden Überblick über den Funktionsumfang und das Handling von WinAdmin/WebAdmin.
  - Das jeweilige **Online-Hilfesystem** beschreibt kontextsensitiv alle Dialogfelder und die zugehörigen Parameter, die die grafische Oberfläche bietet. Außerdem wird dargestellt, wie man WinAdmin bzw. WebAdmin konfiguriert, um stand-alone UTM-Anwendungen und UTM-Cluster-Anwendungen administrieren zu können.

> **i** Details zur Integration von openUTM WebAdmin in den SE Manager des SE Servers finden Sie im SE Server Handbuch **Bedienen und Verwalten**.

Testen und Fehler diagnostizieren

Für die o.g. Aufgaben benötigen Sie außerdem die Handbücher **Meldungen, Test und Diagnose** (jeweils ein Handbuch für Unix-, Linux- und Windows-Systeme und für BS2000-Systeme). Sie beschreiben das Testen einer UTM-Anwendung, den Inhalt und die Auswertung eines UTM-Dumps, das Meldungswesen von openUTM, sowie alle von openUTM ausgegebenen Meldungen und Returncodes.

openUTM-Clients erstellen

Wenn Sie Client-Anwendungen für die Kommunikation mit UTM-Anwendungen erstellen wollen, stehen Ihnen folgende Handbücher zur Verfügung:

- Das Handbuch **openUTM-Client für Trägersystem UPIC** beschreibt Erstellung und Einsatz von Client-Anwendungen, die auf UPIC basieren. Es zeigt auf, was beim Programmieren einer CPI-C-Anwendung zu beachten ist und welche Einschränkungen es gegenüber der Programmschnittstelle X/Open CPI-C gibt.

- Das Handbuch **openUTM-Client für Trägersystem OpenCPIC** beschreibt, wie man OpenCPIC installiert und konfiguriert. Es zeigt auf, was beim Programmieren einer CPI-C-Anwendung zu beachten ist und welche Einschränkungen es gegenüber der Programmschnittstelle X/Open CPI-C gibt.
- Für das mit **BeanConnect** ausgelieferte Produkt **openUTM-JConnect** existiert neben dem Handbuch eine Java-Dokumentation mit der Beschreibung der Java-Klassen.
- Das Handbuch **BizXML2Cobol** beschreibt, wie Sie bestehende Cobol-Programme einer UTM-Anwendung so erweitern können, dass sie als Standard-Web-Service auf XML-Basis genutzt werden können. Die Arbeit mit der grafischen Bedienoberfläche ist in der zugehörigen **Online-Hilfe** beschrieben.
- Sie können auch mit dem Software-Produkt WS4UTM (WebServices for openUTM) Services von UTM-Anwendungen als Web Services verfügbar machen. Dazu benötigen Sie das Handbuch **Web-Services für openUTM.** Die Arbeit mit der grafischen Bedienoberfläche ist in der zugehörigen **Online-Hilfe** beschrieben.

Kopplung mit der IBM-Welt

Wenn Sie aus Ihrer UTM-Anwendung mit Transaktionssystemen von IBM kommunizieren wollen, benötigen Sie außerdem das Handbuch **Verteilte Transaktionsverarbeitung zwischen openUTM und CICS-, IMS- und LU6.2-Anwendungen**. Es beschreibt die CICS-Kommandos, IMS-Makros und UTM-Aufrufe, die für die Kopplung von UTM-Anwendungen mit CICS- und IMS-Anwendungen benötigt werden. Die Kopplungsmöglichkeiten werden anhand ausführlicher Konfigurations- und Generierungsbeispiele erläutert. Außerdem beschreibt es die Kommunikation über openUTM-LU62, sowie dessen Installation, Generierung und Administration.

Dokumentation zu PCMX

Mit openUTM auf Unix-, Linux- und Windows-Systemen wird die Kommunikationskomponente PCMX ausgeliefert. Die Funktionen von PCMX sind in folgenden Dokumenten beschrieben:

- Handbuch CMX (Unix-Systeme) "Betrieb und Administration" für Unix- und Linux- Systeme
- Online-Hilfe zu PCMX für Windows-Systeme

1.2.2 Dokumentation zum openSEAS-Produktumfeld

Die Verbindung von openUTM zum openSEAS-Produktumfeld wird im openUTM-Handbuch **Konzepte und Funktionen** kurz dargestellt. Die folgenden Abschnitte zeigen, welche der openSEAS-Dokumentationen für openUTM von Bedeutung sind.

Integration von Java EE Application Servern und UTM-Anwendungen

Der Adapter BeanConnect gehört zur Produkt-Suite openSEAS. Der BeanConnect-Adapter realisiert die Verknüpfung zwischen klassischen Transaktionsmonitoren und Java EE Application Servern und ermöglicht damit die effiziente Integration von Legacy-Anwendungen in Java-Anwendungen.

Das Handbuch **BeanConnect** beschreibt das Produkt BeanConnect, das einen JCA 1.5- und JCA 1.6-konformen Adapter bietet, der UTM-Anwendungen mit Anwendungen auf Basis von Java EE, z.B. mit dem Application Server von Oracle, verbindet.

Web-Anbindung und Anwendungsintegration

Anstatt der UTM-HTTP-Programmschnittstelle können Sie alternativ auch das Produkt WebTransactions verwenden. Dann benötigen Sie die Handbücher zu WebTransactions. Die Dokumentation wird durch JavaDocs ergänzt.

1.2.3 Readme-Dateien

Funktionelle Änderungen und Nachträge der aktuellen Produktversion zu diesem Handbuch entnehmen Sie bitte ggf. den Produkt-spezifischen Readme-Dateien.

Readme-Dateien stehen Ihnen online bei dem jeweiligen Produkt zusätzlich zu den Produkthandbüchern unter https://bs2manuals.ts.fujitsu.com zur Verfügung. Für die Plattform BS2000 finden Sie Readme-Dateien auch auf der Softbook-DVD.

Informationen auf BS2000-Systemen

Wenn für eine Produktversion eine Readme-Datei existiert, finden Sie auf BS2000-Systemen die folgende Datei:

`SYSRME.<product>.<version>.<lang>`

Diese Datei enthält eine kurze Information zur Readme-Datei in deutscher oder englischer Sprache (<lang>=D/E). Die Information können Sie am Bildschirm mit dem Kommando `/SHOW-FILE` oder mit einem Editor ansehen. Das Kommando `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` zeigt, unter welcher Benutzerkennung die Dateien des Produkts abgelegt sind.

Ergänzende Produkt-Informationen

Aktuelle Informationen, Versions-, Hardware-Abhängigkeiten und Hinweise für Installation und Einsatz einer Produktversion enthält die zugehörige Freigabemitteilung. Solche Freigabemitteilungen finden Sie online unter https://bs2manuals.ts.fujitsu.com.

1.3 Änderungen in openUTM V7.0

Die folgenden Abschnitte gehen näher auf die Änderungen in den einzelnen Funktionsbereichen ein.

1.3.1 Neue Server-Funktionen

UTM als HTTP-Server

Eine UTM-Anwendung kann auch als HTTP-Server fungieren.

Als Methoden werden GET, PUT, POST und DELETE unterstützt. Neben HTTP wird auch der Zugang über HTTPS unterstützt.

Dazu wurden folgende Schnittstellen geändert:

- Generierung

  Alle Systeme:

  - KDCDEF-Anweisung BCAMAPPL:
    - Beim Operand T-PROT= mit Wert SOCKET gibt es eine zusätzliche Angabe zum Transportprotokoll:

      *USP: Auf Verbindungen dieses Zugriffspunktes soll das UTM-Socket-Protokoll verwendet werden.

      *HTTP: Auf Verbindungen dieses Zugriffspunktes soll das HTTP-Protokoll verwendet werden.

      *ANY: Auf Verbindungen dieses Zugriffspunktes werden sowohl das UTM-Socket-Protokoll als auch das HTTP-Protokoll unterstützt.

    - Beim Operand T-PROT= mit Wert SOCKET gibt es zusätzlich die Angabe zur Verschlüsselung:

      SECURE: Auf Verbindungen dieses Zugriffspunktes erfolgt die Kommunikation unter Verwendung von Transport Layer Security (TLS).

    - Neuer Operand USER-AUTH = *NONE | *BASIC. Hiermit kann angegeben werden, welchen Authentisierungsmechanismus HTTP-Clients für diesen Zugangspunkt verwenden müssen.
  - KDCDEF-Anweisung HTTP-DESCRIPTOR:
    Mit dieser Anweisung wird eine Abbildung des in einem HTTP-Request empfangenen Path auf einen TAC definiert und es können zusätzliche Verarbeitungsparameter angegeben werden.

  BS2000-Systeme:

  - KDCDEF-Anweisung CHAR-SET:
    Mit dieser Anweisung können jeder der von openUTM zur Verfügung gestellten vier Code-Konvertierungen von UTM jeweils bis zu vier Character-Set Namen zugeordnet werden.
- Programmierung
  - KDCS- Kommunikationsbereich (KB):
    Im Kopf des KDCS-Kommunikationsbereichs gibt es im Feld *kccp/KCCP* neue Werte für die Client-Protokolle HTTP, USP-SECURE und HTTPS.
  - KDCS-Aufruf INIT PU:
    - Die Version der Schnittstelle wurde auf 7 erhöht.
    - Um die verfügbare Information vollständig zu erhalten, muss im Feld KCLI der Wert 372 angegeben werden.
    - Neue Felder zur Anforderung (KCHTTP/http_info) und Rückgabe (KCHTTPINF/httpInfo) von HTTP-spezifischen Informationen.
- Administrationsschnittstelle KDCADMI
  - Die Datenstrukturversion von KDCADMI wurde auf Version 11 geändert (Feld *version_data* im Parameterbereich).

- Neue Struktur *kc_http_descriptor_str* im Identifikationsbereich für die Unterstützung des HTTP Deskriptors.
- Neue Struktur *kc_character_set_str* im Identifikationsbereich für die Unterstützung des HTTP Charactersets.
- Neue Felder *secure_soc* und *user_auth* in Struktur *kc_bcamappl_str* für die Unterstützung von HTTP Zugangspunkten.

- Programmschnittstelle UTM-HTTP
  Zusätzlich zum KDCS-Interface bietet UTM ein Interface zum Lesen und Schreiben von HTTP Protokollinformationen und zur Behandlung des HTTP Message Bodys.
  Im Folgenden werden die Funktionen des Interface kurz aufgelistet:

- Funktion *kcHttpGetHeaderByIndex()*
  Diese Funktion liefert den Namen und Wert des HTTP-Header-Feldes für den angegebenen Index zurück.
- Funktion *kcHttpGetHeaderByName()*
  Die Funktion liefert den Wert des über den Namen spezifizierten HTTP-Header-Feldes zurück.
- Funktion *kcHttpGetHeaderCount()*
  Diese Funktion liefert die Anzahl der in dem HTTP-Request enthaltenen Header-Felder zurück, die vom Teilprogramm gelesen werden können .
- Funktion *kcHttpGetMethod()*
  Diese Funktion liefert die HTTP-Methode des HTTP-Requests zurück.
- Funktion *kcHttpGetMputMsg()*
  Diese Funktion liefert die vom Teilprogramm erzeugte MPUT-Nachricht zurück.
- Funktion *kcHttpGetPath()*
  Diese Funktion liefert den mit KC_HTTP_NORM_UNRESERVED normierten HTTP- Path des HTTP-Requests zurück.
- Funktion *kcHttpGetQuery()*
  Diese Funktion liefert die mit KC_HTTP_NORM_UNRESERVED normierte HTTP- Query des HTTP -Requests zurück.
- Funktion *kcHttpGetRc2String()*
  Hilfsfunktion um ein Funktionsergebnis vom Typ enum in einen abdruckbaren null-terminierten String umzuwandeln.
- Funktion *kcHttpGetReqMsgBody()*
  Diese Funktion liefert den Message Body des HTTP Requests zurück.
- Funktion *kcHttpGetScheme()*
  Diese Funktion liefert das Schema des HTTP- Requests zurück.
- Funktion *kcHttpGetVersion()*
  Diese Funktion liefert die Version des HTTP- Requests zurück .
- Funktion *kcHttpPercentDecode()*
  Funktion zur Umwandlung von Zeichen in Prozent-Darstellung in Zeichenfolgen in normale Ein-Zeichen-Darstellung.
- Funktion *kcHttpPutHeader()*
  Diese Funktion übergibt einen HTTP-Header für die HTTP-Response .
- Funktion *kcHttpPutMgetMsg()*
  Diese Funktion übergibt eine Nachricht für das Teilprogramm, die mit MGET gelesen werden kann.
- Funktion *kcHttpPutRspMsgBody()*
  Diese Funktion übergibt eine Nachricht für den Message Body der HTTP-Response.
- Function *kcHttpPutStatus()*
  Diese Funktion übergibt einen HTTP-Statuscode für die HTTP-Response.

- Kommunikation über den Secure Socket Layer (SSL)
  BS2000-Systeme:
  - Ist für eine UTM-Anwendung ein BCAMAPPL mit T-PROT=(SOCKET,...,SECURE) generiert, dann wird beim Anwendungsstart von UTM eine zusätzliche Task mit einem Reverse Proxy gestartet, der für die Anwendung als TLS Termination Proxy fungiert und über den sämtliche SSL-Kommunikation abgewickelt wird.

Unix-, Linux- und Windows-Systeme :

- Für einen sicheren Zugang über TLS steht ein weiterer Netzprozess vom Typ *utmnetssl* zur Verfügung. Sind für eine UTM-Anwendung BCAMAPPL mit T-PROT=(SOCKET,...,SECURE) generiert, dann wird beim Anwendungsstart von UTM eine Anzahl von *utmnetssl* Prozessen gestartet. Die Anzahl dieser Prozesse ist abhängig vom Wert LISTENER-ID dieser BCAMAPPL Objekte. In einem *utmnetssl* Prozess wird für die zugeordneten BCAMAPPL Portnummern die gesamte TLS-Kommunikation abgewickelt.

Verschlüsselung

Die Verschlüsselungsfunktionalität in UTM zwischen einer UTM-Anwendung und einem UPIC-Client wurde überarbeitet. Dabei wurden Sicherheitslücken geschlossen, moderne Methoden aufgenommen und die Auslieferung wie folgt vereinfacht:

- UTM-CRYPT Variante
  Bisher stand die Verschlüsselungsfunktionalität in UTM nur zur Verfügung, wenn man das Produkt UTM-CRYPT installiert hatte. Mit UTM V7.0 ist dies nicht mehr erforderlich. Ab dieser Version wird über die Generierung bzw. zum Anwendungsstart entschieden, ob die Verschlüsselungsfunktionalität zum Einsatz kommt oder nicht.
- Security
  Bei der Kommunikation zwischen einer UTM-Anwendung und einem UPIC-Client wurde eine Sicherheitslücke behoben.

  > **!** Das hat zur Folge, dass verschlüsselte Kommunikation einer UTM-Anwendung V7.0 nur zusammen mit UPIC-Client Anwendungen ab UPIC V7.0 möglich ist!

- Verschlüsselung Level 5 (*Unix-, Linux- und Windows-Systeme):*
  KDCDEF-Anweisungen PTERM, TAC und TPOOL
  Beim Operanden ENCRYPTION-LEVEL gibt es einen zusätzlichen Level 5. Dabei wird zur Vereinbarung des Session-Keys das auf Elliptic Curves basierende Diffie-Hellman Verfahren verwendet und Ein-/Ausgabe-Nachrichten werden mit dem AES-GCM Algorithmus verschlüsselt.

OSI-TP Kommunikation und Portnummern

BS2000-Systeme:

- KDCDEF-Anweisung OSI-CON
  Der Operand LISTENER-PORT kann auch auf BS2000-Systemen angegeben werden.
- Administrationsschnittstelle KDCADMI
  In der Struktur *kc_osi_con_str* wird auch auf BS2000-Systemen im Feld *listener-port* die Portnummer angezeigt.

Subnetze

In einer UTM-Anwendung können auch auf BS2000-Systemen Subnetze generiert werden, um den Zugang zu UTM-Anwendungen auf definierte IP-Adressbereiche beschränken zu können. Zusätzlich kann die Namensauflösung per DNS gesteuert werden.

Dazu wurden folgende Schnittstellen geändert:

- Generierung
  BS2000-Systeme:
  - KDCDEF-Anweisung SUBNET:
    Die SUBNET-Anweisung kann auch auf BS2000-Systemen angegeben werden.

  Alle Systeme:
  - KDCDEF-Anweisung SUBNET:
    Mit RESOLVE-NAMES=YES/NO kann angegeben werden, ob nach einem Verbindungsaufbau eine Namensauflösung per DNS stattfinden soll oder nicht.

    Falls eine Namensauflösung erfolgt, dann wird über die Administrationsschnittstelle und in Meldungen der echte Prozessorname des Kommunikationspartners angezeigt. Andernfalls wird als Prozessorname die IP-Adresse der Komunikationspartners sowie der Name des in der Generierung definierten Subnetzes angezeigt.
- Administrationsschnittstelle KDCADMI
  Die Strukturen *kc_subnet_str* und *kc_tpool_str* enthalten ein neues Feld *resolve_names*.

Zugangsdaten für den XA-Datenbank-Anschluss

Ein modifizierter aber noch nicht aktivierter Benutzername für den XA-Datenbank-Anschluss kann per Administration (KDCADMI) gelesen werden:

- Operationscode KC_GET_OBJECT:
  Datenstruktur *kc_db_info_str*: Neues Feld *db_new_userid*.

Reconnect für den XA-Datenbank-Anschluss

Wird bei einer XA Aktion zur Steuerung der Transaktion entdeckt, dass die Verbindung zur Datenbank nicht mehr besteht, wird versucht die Verbindung zu erneuern und die XA Aktion zu wiederholen.

Nur falls dies nicht erfolgreich ist, werden der betroffene UTM Prozess und die UTM-Anwendung abnormal beendet . Bisher wurde bei jedem Verbindungsverlust zur XA Datenbank unmittelbar ohne erneuten Verbindungs-Versuch die UTM-Anwendung abnormal beendet.

Sonstige Änderungen

- XA-Meldungen
  Die Meldungen bzgl. der XA-Schnittstelle wurden jeweils um die Inserts UTM-Userid und TAC erweitert. Betroffen sind die Meldungen K204-K207, K212-K215 und K217-K218.
- UTM-Tool KDCEVAL
  Im TRACE 2 Satz von KDCEVAL wurde im WAITEND Record der Typ des letzten Auftrags (Börsen-Announcements) aufgenommen (ersten beiden Bytes abdruckbar).

1.3.2 Entfallene Server-Funktionen

Im Einzelnen wurden folgende Funktionen gestrichen:

- Dienstprogramm KDCDEF
  Mehrere Funktionen wurden gestrichen und können nicht mehr in KDCDEF generiert werden. Wenn sie dennoch angegeben werden, wird dies im KDCDEF-Lauf mit einem Syntaxfehler abgelehnt.
  - KDCDEF-Anweisung PTERM
    Operanden-Werte 1 und 2 für ENCRYPTION-LEVEL
  - KDCDEF-Anweisung TPOOL
    Operanden-Werte 1 und 2 für ENCRYPTION-LEVEL
  - KDCDEF-Anweisung TAC
    Operanden-Wert 1 für ENCRYPTION-LEVEL
- *BS2000-Systeme*
  - UTM-Cluster:
    Auf BS2000-Systemen werden UTM-Cluster-Anwendungen nicht mehr unterstützt.
- *Unix-, Linux- und Windows-Systeme*
  - TNS Betrieb:
    Beim Start einer UTM-Anwendung wird die TNS-Generierung nicht mehr gelesen. Die Adressierungsinformation muss vollständig bei der Konfiguration mit KDCDEF hinterlegt werden.

1.3.3 Neue Client-Funktionen

Verschlüsselung

Die Verschlüsselungsfunktionalität in openUTM-Client wurde überarbeitet. Dabei wurden Sicherheitslücken geschlossen, moderne Methoden aufgenommen und die Auslieferung wie folgt vereinfacht:

- UTM-CLIENT-CRYPT Variante
  Bisher stand die Verschlüsselungsfunktionalität in openUTM-Client nur zur Verfügung, wenn man das Produkt UTM-CLIENT-CRYPT installiert hatte. Mit openUTM-Client V7.0 ist dies nicht mehr erforderlich. Ab dieser Version wird zum Ablaufzeitpunkt entschieden ob die Verschlüsselungsfunktionalität zum Einsatz kommt oder nicht.
- Security
  Bei der Kommunikation mit einer UTM-Anwendung wurde eine Sicherheitslücke behoben.
- Verschlüsselung Level 5
  openUTM-Client V7.0 unterstützt die Kommunikation mit UTM V7.0 Anwendungen, bei denen für die Verbindungen zum UPIC-Client ENCRYPTION-LEVEL 5 generiert wurde.
  Bei Level 5 wird zur Vereinbarung des Session-Keys das auf Elliptic Curves basierende Diffie-Hellman Verfahren verwendet und Ein-/Ausgabe-Nachrichten werden mit dem AES-GCM Algorithmus verschlüsselt. AES-GCM unterstützt die Authentifikation und die Verschlüsselung von Nachrichten.

  Der Level 5 wird von openUTM-Client auf allen Plattformen unterstützt.
- Verschlüsselung BS2000
  openUTM-Client (BS2000) verwendet analog zu Unix-, Linux- und Windows-Systemen openSSL anstatt BS2000-CRYPT.

1.3.4 Neue Funktionen für openUTM WinAdmin

WinAdmin unterstützt alle Neuerungen der openUTM V7.0 bzgl. der Programmschnittstelle zur Administration.

1.3.5 Neue Funktionen für openUTM WebAdmin

WebAdmin unterstützt alle Neuerungen der openUTM V7.0 bzgl. der Programmschnittstelle zur Administration.

1.4 Darstellungsmittel

Metasyntax

Die in diesem Handbuch verwendete Metasyntax können Sie der folgenden Tabelle entnehmen:

Formale Darstellung	Erläuterung	Beispiel
GROSSBUCHSTABEN	Großbuchstaben bezeichnen Konstanten (Namen von Aufrufen, Anweisungen, Feldnamen, Kommandos und Operanden etc.), die in dieser Form anzugeben sind.	`LOAD-MODE=STARTUP`
kleinbuchstaben	In Kleinbuchstaben sind in Syntaxdiagrammen und Operandenbeschreibung die Platzhalter für Operandenwerte dargestellt.	`KDCFILE=filebase`
kleinbuchstaben	Im Fließtext werden Variablen sowie Namen von Datenstrukturen und Feldern in kursiven Kleinbuchstaben dargestellt.	*utm-installationsverzeichnis* ist das UTM-Installationsverzeichnis
`Schreibmaschinenschrift`	In Schreibmaschinenschrift werden im Fließtext Kommandos, Dateinamen, Meldungen und Beispiele ausgezeichnet, die in genau dieser Form eingegeben werden müssen bzw. die genau diesen Namen oder diese Form besitzen.	Der Aufruf `tpcall`
{ } und |	In geschweiften Klammern stehen alternative Angaben, von denen Sie eine auswählen müssen. Die zur Verfügung stehenden Alternativen werden jeweils durch einen Strich getrennt aufgelistet.	`STATUS={ ON | OFF }`
[ ]	In eckigen Klammern stehen wahlfreie Angaben, die entfallen können.	`KDCFILE=( filebase [, { SINGLE | DOUBLE } ] )`
( )	Kann für einen Operanden eine Liste von Parametern angegeben werden, sind diese in runde Klammern einzuschließen und durch Kommata zu trennen. Wird nur ein Parameter angegeben, kann auf die Klammern verzichtet werden.	`KEYS=(key1, key2,...key`*n*`)`
Unterstreichen	Unterstreichen kennzeichnet den Standardwert.	`CONNECT= { YES | NO }`
Kurzform	Die Standardkurzform für Anweisungen, Operanden und Operandenwerte wird „fett“ hervorgehoben. Die Kurzform kann alternativ angegeben werden.	`TRANSPORT -SELECTOR =c‘C‘`

Formale Darstellung	Erläuterung	Beispiel

`...`	Punkte zeigen die Wiederholbarkeit einer syntaktischen Einheit an.
	Außerdem kennzeichnen die Punkte Ausschnitte aus einem Programm, einer Syntaxbeschreibung o.ä.
		`KDCDEF starten`
		`...`
		`OPTION`
		`DATA=statement_file`
		`...`
		`END`

Symbole

 für Verweise auf umfassende und detaillierte Informationen zum jeweiligen Thema.

 für Hinweistexte.

 für Warnhinweise.

Sonstiges

utmpfad bezeichnet auf Unix-, Linux- und Windows-Systemen das Verzeichnis, unter dem openUTM installiert wurde.

filebase bezeichnet auf Unix-, Linux- und Windows-Systemen das Dateiverzeichnis der UTM-Anwendung. Dies ist der Basisname, der in der KDCDEF-Anweisung MAX KDCFILE= generiert wurde.

$userid bezeichnet auf BS2000-Systemen die Kennung, unter der openUTM installiert wurde.

upic-dir bezeichnet auf Unix-, Linux- und Windows-Systemen das Verzeichnis, unter dem openUTM-Client für Trägersystem UPIC installiert ist.

2 Struktur des Anwendungsprogramms festlegen

In diesem Kapitel wird auf die Strukturierung eines Anwendungsprogramms eingegangen.

Bitte beachten Sie, dass Sie das Anwendungsprogramm mit dem Dienstprogramm BINDER binden und über die Funktionen des BLS laden müssen.

Ein UTM-Anwendungsprogramm kann unterschiedlich strukturiert und auf verschiedene Weise geladen werden. Generell lassen sich fünf verschiedene Bereiche unterscheiden:

- Shareable Programme im Systemspeicher.

  Programmteile, die in den Systemspeicher geladen werden, stehen allen Prozessen eines BS2000-Systems gemeinsam zur Verfügung. Daher sollten Sie in den Systemspeicher in erster Linie Programme laden, die Anwendungs-unabhängig sind, wie z.B. die shareable Teile von Laufzeitsystemen der Programmiersprachen. Sehen Sie dazu Abschnitt „Shared Code im Systemspeicher“.

  Die shareable Module müssen vom Systemadministrator als nicht-privilegiertes Subsystem geladen werden.

- Statisch gebundene Programmteile.

  Die ROOT-Systemmodule sowie die von diesen benötigten Module des Laufzeitsystems (ILCS) und die Meldungsmodule, für die in der UTM-Generierung keine Nachladebibliothek angegeben wurde, müssen Sie statisch in das Start-LLM einbinden. Das von KDCDEF erzeugte und anschließend von Ihnen assemblierte ROOT-Tabellenmodul kann statisch eingebunden oder beim Start dynamisch nachgeladen werden.

  Auch weitere Teile der Anwendung können Sie in den statischen Teil einbinden, jedoch dürfen diese keine Externverweise auf Module enthalten, die Sie in einen von openUTM verwalteten Common Memory Pool laden.

  Den statischen Teil einer UTM-Anwendung laden Sie mit dem Kommando

  `START-EXECUTABLE-PROGRAM` bzw. `LOAD-EXECUTABLE-PROGRAM`

- Shareable Programme in Common Memory Pools.

  Programmteile, die von allen Prozessen einer UTM-Anwendung gemeinsam benutzt werden können, wie z.B. die shareable Teile der Teilprogramme der UTM-Anwendung oder auch Formate oder Datenbereiche, sollten Sie in Common Memory Pools laden. Sehen Sie dazu Abschnitt „Shared Code in Common Memory Pools“.

- Beim Start der Anwendung zu ladende Programmteile.

  Programmteile, die ständig von der UTM-Anwendung benötigt werden oder die Externverweise zu shareable Teilen der Anwendung enthalten, sollten Sie beim Anwendungsstart dynamisch nachladen lassen.

  Auch das vom KDCDEF erzeugte und assemblierte ROOT-Tabellenmodul wird dynamisch nachgeladen, falls Sie es nicht statisch zum Start-LLM dazugebunden haben.

- Beim Programmaufruf zu ladende Programmteile.

  Programmteile, die nicht ständig von der UTM-Anwendung benötigt werden, können Sie so generieren, dass diese erst zum Zeitpunkt des ersten Aufrufs nachgeladen werden. Diese Programmteile müssen Sie derart zu LLMs vorbinden, dass alle offenen Externverweise beim Laden aus den bereits im Speicher vorhandenen Modulen befriedigt werden können.

In welchen der Speicherbereiche und zu welchem Zeitpunkt die Programmteile geladen werden, ist abhängig von den Angaben, die Sie bei der UTM-Generierung der Anwendung mit KDCDEF machen (siehe openUTM-Handbuch „Anwendungen generieren“).

Informationen dazu finden Sie im Handbuch in den folgenden Kapiteln:

- Kapitel „Anwendungsprogramm erzeugen“
- Kapitel „Programmaustausch im Betrieb“

2.1 Lademodule generieren

Nur ein Teil der Anwendung muss statisch zum Anwendungsprogramm gebunden werden (Start-LLM, siehe Abschnitt „Binden des Anwendungsprogramms“). Die anderen Teile des Anwendungsprogramms müssen Sie in Form von dynamisch nachladbaren Lademodulen bereitstellen.

Bei den Lademodulen kann es sich um einzelne Objektmodule (OM) handeln, die in einer Bindemodulbibliothek (OML) oder einer Programmbibliothek als Elemente vom Typ R bereitgestellt werden, oder um Bindelademodule (LLM), die in einer Programmbibliothek als Elemente vom Typ L bereitgestellt werden. Gruppen von Lademodulen können Sie mit dem BINDER zu LLMs vorbinden, siehe Abschnitt „Binden von LLMs“.

Bei der KDCDEF-Generierung müssen die einzelnen Lademodule der Anwendung mit LOAD-MODULE-Anweisungen generiert werden.

Im einzelnen legen Sie bei der KDCDEF-Generierung für die Lademodule Folgendes fest:

- wann die Lademodule geladen werden sollen

  Für Teilprogramme, d.h. Module, die mit der PROGRAM-Anweisung generiert wurden, können Sie zwischen dem Startzeitpunkt der Anwendung und dem Aufrufzeitpunkt des Teilprogramms wählen.

  Nicht-shareable Teilprogramme können entweder statisch zum Anwendungsprogramm gebunden, als Lademodul beim Anwendungsstart dynamisch nachgeladen oder als Lademodul zum Zeitpunkt des Teilprogrammaufrufs geladen werden.

  Der nicht-shareable Teil (Private Slice) zu einem shareable Teil (Public Slice), der in einem Common Memory Pool steht, kann ebenfalls entweder beim Anwendungsstart oder erst zum Zeitpunkt des Teilprogrammaufrufs geladen werden.

  Der Private Slice zu einem Public Slice, der sich in nichtprivilegierten Subsystemen befindet, kann man alternativ auch zum statischen Teil des Anwendungsprogramms binden.

  Module und Datenbereiche, die mit der AREA-Anweisung generiert wurden, müssen entweder statisch zum Anwendungsprogramm gebunden oder zum Zeitpunkt des Starts der Anwendung nachgeladen werden, da der Zugriff auf diese Anwendungsteile nicht unter der Kontrolle von openUTM erfolgt.

- aus welchen Bibliotheken die Lademodule geladen werden sollen

  Sie können Bindemodulbibliotheken (OML) oder Programmbibliotheken (PL), die Elemente vom Typ R oder L enthalten, angeben.

- welche Version eines Lademoduls geladen werden soll

  In einer Programmbibliothek können gleichzeitig mehrere Versionen eines Elements enthalten sein. Mit der Versionsangabe legen Sie fest, welche Version eines Elements geladen werden soll.

- in welchen Speicherbereich die Lademodule geladen werden sollen

  Ein Lademodul kann entweder in einen Common Memory Pool oder in den Standardkontext der Anwendung geladen werden. Der Standardkontext enthält den statisch gebundenen Teil des Anwendungsprogramms, der mit dem Kommando START-EXECUTABLE-PROGRAM bzw. LOAD-EXECUTABLE-PROGRAM geladen wurde, sowie die Teile des Anwendungsprogramms, die nicht in Common Memory Pools oder in den Systemspeicher geladen wurden.

  Enthält ein LLM Public und Private Slices, dann kann der Public Slice in einen Common Memory Pool und der Private Slice in den Standardkontext im task-lokalen Speicher geladen werden.

- ob mit Autolink gebunden werden soll

  Die shareable Teile der Lademodule werden stets ohne die Autolink-Funktion geladen. Bei den nicht-shareable Teilen können Sie über die LOAD-MODULE-Anweisung steuern, ob mit oder ohne Autolink-Funktion geladen werden soll.

  Wird die Autolink-Funktion des BLS beim Nachladen und beim Programmaustausch unterdrückt, dann werden der Ladevorgang für Lademodule beschleunigt und Inkonsistenzen in dem geladenen Anwendungsprogramm vermieden. In diesem Fall dürfen die Lademodule ausschließlich offene Externverweise auf Programmteile besitzen, die zum Zeitpunkt des Ladens dieses Moduls bereits im Arbeitsspeicher vorhanden sind.

  Wird mit Autolink geladen, dann werden beim Laden des Lademoduls die Module, die zusätzlich benötigt werden, nachgebunden. Die Autolink-Funktion sollte nur für Module des Laufzeitsystems, *nicht* jedoch für benutzereigene Module verwendet werden, da per Autolink geladene Module bei einem nachfolgenden Austausch nicht mehr entladen werden.

Die Zuordnung von Objekten (Teilprogramme und gemeinsam nutzbare Datenbereiche) zu den Lademodulen wird ebenfalls bei der UTM-Generierung festgelegt (Operand LOAD-MODULE in den PROGRAM- und AREA-Anweisungen). Ob die mit KDCDEF definierte Zuordnung der tatsächlichen Aufteilung der Lademodule in den Bibliotheken entspricht, kann von openUTM nicht verifiziert werden. openUTM verlässt sich beim Programmaustausch auf die bei der UTM-Generierung gemachten Angaben.

Mit der Reihenfolge, mit der Sie die Lademodule generieren, bestimmen Sie die Reihenfolge, in der die Lademodule geladen werden. Wenn Sie Lademodule, die mit LOAD-MODE=(POOL, poolname, STARTUP) oder LOAD-MODE=STARTUP generiert wurden, ohne die Autolink-Funktion des BLS laden lassen, ist die Reihenfolge der UTM-Generierung ausschlaggebend für die Auflösung der unbefriedigten Externverweise beim Laden.

Ausführliche Informationen zum Generieren der Lademodule finden Sie im
openUTM-Handbuch „Anwendungen generieren“.

Hinweise zum Programmaustausch

Mit der Generierung der Lademodule legen sie fest, welche Programmteile später ausgetauscht werden können. Folgendes ist zu beachten:

- Programmteile, die später ausgetauscht werden sollen, dürfen nicht statisch ins Anwendungsprogramm eingebunden oder in anwendungsglobale Common Memory Pools oder in den Systemspeicher geladen werden.
- Das BLS unterstützt nur das Entladen von Programmteilen, die mit **einem** Ladevorgang geladen werden. Wenn also eine Gruppe von Modulen ausgetauscht werden soll, müssen Sie diese Module als LLM oder OM vorbinden. Soll ein Einzelmodul ausgetauscht werden, können Sie dieses Modul als LLM oder OM bereitstellen. Es darf dann nicht mit anderen Modulen zusammen gebunden werden.
- Ein Lademodul, das das Administrationsprogramm KDCADM, die Event-Exits START, SHUT, INPUT und FORMAT oder die Event-Services BADTAC, MSGTAC und SIGNON enthält, darf im Betrieb nicht ausgetauscht werden.
- Bei einer Änderungsgenerierung werden standardmäßig auch die aktuellen Versionsnummern der ausgetauschten Lademodule aus der alten KDCFILE in die neue KDCFILE übertragen.

2.2 Module laden

In welcher Reihenfolge die verschiedenen Programmteile der UTM-Anwendung geladen und wie dabei die Externverweise aufgelöst werden, wird in Bild 1 dargestellt und anschließend beschrieben. Die Nummern bezeichnen dabei die Reihenfolge, in der die Programmteile geladen werden (müssen). Die Pfeile geben an, in welcher Richtung beim Laden die unbefriedigten Externverweise der Lademodule aufgelöst werden, wenn ohne Autolink geladen wird.

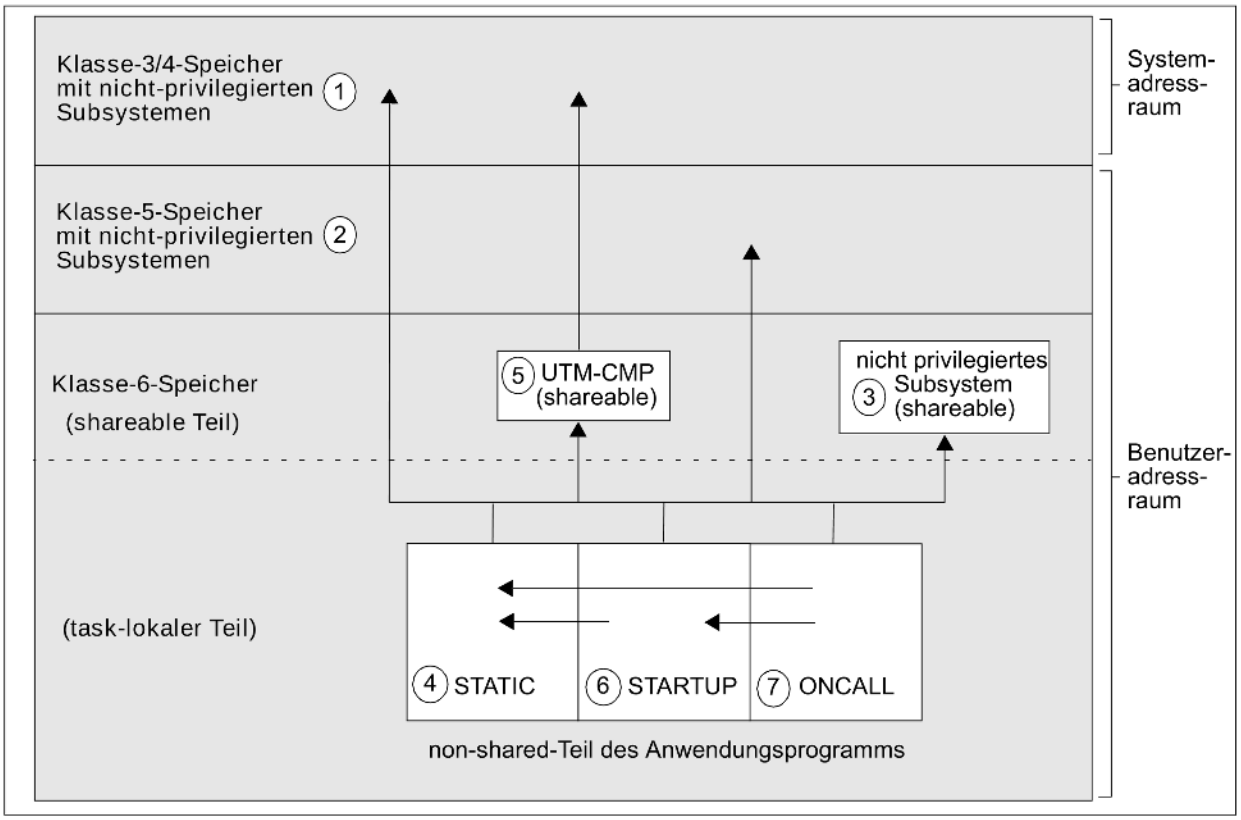


Bild 1: Speicherstruktur einer UTM-Anwendung (CMP = Common Memory Pool)

Mit Ausnahme des Ladens der Subsysteme (durch den Systemadministrator) und des Ladens des statischen Teils (durch ein START-EXECUTABLE-PROGRAM-Kommando) werden alle Ladevorgänge durch openUTM angestoßen und kontrolliert:

1, 2, 3 Gemeinsam benutzbare Module lädt der Systemadministrator vor dem Start der UTM-Anwendung.

4 Das Start-LLM mit allen eingebundenen Lademodulen wird mit dem START-EXECUTABLE-PROGRAM- oder LOAD-EXECUTABLE-PROGRAM-Kommando geladen. Offene Externverweise sollten wegen der Ladeperformance nur aus dem Shared Code befriedigt werden, können aber auch aus Bibliotheken befriedigt werden.

   Das Kommando zum Start des Start-LLMs sollte folgendermaßen lauten:

```
/START-EXECUTABLE-PROGRAM FROM-FILE=*LIB-ELEM                        - *)
/     (LIBRARY=lm-lib                                                -
/     ,ELEMENT=lm-name, TYPE=L)                                      -
/     ,DBL-PAR=(LOADING=                                             -
/                 (LOAD-INFORMATION=REFERENCES,PROGRAM-MODE=ANY)    -
/              ,RESOLUTION=(ALTERNATE-LIBRARIES=*BLSLIB##           -
/                          ,AUTOLINK=*ALTERNATE-LIBRARIES)          -
/              ,ERROR-PROCESSING=(UNRESOLVED-EXTRNS=*DELAY))
*) oder LOAD-EXECUTABLE-PROGRAM
```

   Die Parameter UNRESOLVED-EXTRNS=*DELAY und LOAD-INFORMATION=*REFERENCES sind Pflicht für den Start von UTM-Anwendungen.

   Über AUTOLINK=*ALTERNATE-LIBRARIES wird eingestellt, dass für die Autolink-Funktion nur die alternativen Bibliotheken herangezogen werden und über ALT-LIB=*BLSLIB## wird eingestellt, dass als alternative Bibliotheken nur die mit Linknamen BLSLIBnn verwendet werden.

5 Alle Lademodule, die mit LOAD-MODE=POOL und mit einem Common Memory Pool mit SCOPE=GLOBAL generiert wurden, werden in der Reihenfolge der MPOOL-Anweisungen geladen.

   Danach werden alle Lademodule geladen, die mit LOAD-MODULE=POOL und mit einem Common Memory Pool mit SCOPE=GROUP generiert wurden.

   Innerhalb eines Pools werden die Lademodule in der Reihenfolge der LOAD-MODULE-Anweisungen zu diesem Pool geladen.

   Beim Laden werden Externverweise nur aus dem Systemspeicher und aus dem eigenen Memory Pool befriedigt; beim Laden von Shared Code wird die Autolink-Funktion unterdrückt.

6 Die Lademodule, die mit LOAD-MODE=STARTUP generiert wurden, werden beim Start des Anwendungsprogramms dynamisch nachgeladen. Die Reihenfolge der LOAD-MODULE-Anweisungen bei der UTM-Generierung bestimmt auch hier die Reihenfolge beim Laden.

   Offene Externverweise können aus dem Systemspeicher, aus den Common Memory Pools und den bis dahin bereits geladenen Modulen befriedigt werden. Externverweise auf die Laufzeitsysteme können durch Nachladen von Modulen der Laufzeitsysteme befriedigt werden (Angabe von ALTERNATE-LIBRARIES=YES in der LOAD-MODULE-Anweisung).
   Das gleiche gilt für das Laden der Private Slices von Lademodulen, die mit LOAD-MODE=(POOL,..., STARTUP | ONCALL) generiert wurden.

7 Lademodule, die mit LOAD-MODE=ONCALL generiert wurden, werden beim erstmaligen Aufruf eines zugeordneten Teilprogramms geladen. Offene Externverweise werden wie bei den mit LOAD-MODE=STARTUP generierten Lademodulen (siehe 6) aufgelöst.

2.3 Empfehlungen für die Strukturierung der Anwendung

Die Ladestruktur einer Anwendung bestimmt maßgeblich die Zeit, die zum Starten der ersten Task und der Folgetasks einer Anwendung benötigt wird. Ebenso legen Sie damit fest, welche Programmteile später während des Anwendungslaufs ausgetauscht werden können. Deshalb sollten Sie folgende Empfehlungen beachten:

- Der mit dem Kommando START-EXECUTABLE-PROGRAM zu ladende statisch gebundene Teil des Anwendungsprogramms (Start-LLM) sollte möglichst klein gehalten werden. Er muss die ROOT-Systemmodule und die von diesen Modulen benötigten Laufzeitmodule enthalten.
- Das Administrationsprogramm KDCADM muss zum Start der Anwendung zur Verfügung stehen. Es muss daher entweder statisch zum Start-LLM oder beim Start der Anwendung nachgeladen werden. Im zweiten Fall muss es entweder direkt mit LOAD-MODULE KDCADM,...,LOAD-MODE=STARTUP generiert werden oder in ein Lademodul eingebunden werden, das mit LOAD-MODE=STARTUP definiert ist. Selbstgeschriebene Administrationsprogramme sollten wie das Lademodul KDCADM beim Start der Anwendung geladen werden, also wie dieses statisch zum Start-LLM gebunden oder beim Start der Anwendung nachgeladen werden. Ein Lademodul, das das Administrationsprogramm KDCADM enthält, darf im Betrieb nicht ausgetauscht werden. Daher dürfen Sie selbstgeschriebene Administrations-Programme, die austauschbar sein sollen, nicht mit dem KDCADM zusammenbinden.
- Die Event-Exits START, SHUT, INPUT und FORMAT und die Event-Services BADTAC, MSGTAC und SIGNON sollten entweder statisch zum Start-LLM oder zu einem eigenen Lademodul gebunden werden. Dieses Lademodul muss beim Start der Anwendung geladen werden, d.h. es muss mit LOAD-MODULE ...,LOAD-MODE=STARTUP generiert werden. Es sollte im Betrieb nicht ausgetauscht werden, da Fehler beim Austausch dieses Moduls zu einem abnormalen Anwendungsende führen können.
- Programmteile, die später ausgetauscht werden sollen, dürfen nicht statisch gebunden oder in anwendungsglobale Common Memory Pools oder in den Systemspeicher geladen werden.
- Programmteile, die von mehr als einer Anwendung benötigt werden, sollten shareable, z.B. als nicht-privilegiertes Subsystem, geladen werden.
- Alle übrigen Anwendungsteile, die shareable sind, sollten Sie zu einem Lademodul vorbinden und diesen in einen Common Memory Pool laden. Werden aus bestimmten Gründen mehrere Lademodule mit shareable Programmteilen benötigt, so sollten diese trotzdem nur in **einen** Common Memory Pool geladen werden.
- Viele Compiler erlauben es, das shareable Modul mit dem nicht-shareable Modul gemeinsam in einem LLM abzulegen. Dies vereinfacht die Verwaltung der Module, da sich beide Teile immer in einem Container befinden. In diesem Fall sollten Sie die shareable und die nicht-shareable Teile nicht getrennt vorbinden, sondern die LLMs mit Public und Private Slices, wie sie vom Compiler erzeugt wurden, in ein oder mehrere LLMs vorbinden.

  openUTM lädt den Public Slice eines solchen LLM in einen Memory Pool und den Private Slice beim Start der Anwendung bzw. beim Aufruf eines Programms aus einem solchen LLM in den task-lokalen Speicher, wenn Sie das LLM mit LOAD-MODULE ..., LOAD-MODE=(POOL,...,STARTUP) bzw. LOAD-MODE= (POOL,...,ONCALL) generieren.
- Programmteile, die nur gelegentlich benötigt werden, sollten so generiert werden, dass sie erst zum Aufrufzeitpunkt nachgeladen werden
  (LOAD-MODULE ..., LOAD-MODE= ONCALL). Dadurch kann die Startphase verkürzt werden. Diese Lademodule sollten Sie zu kleinen logischen Einheiten vorbinden, da das Laden kleiner Module weniger Zeit benötigt als das Laden großer Module.
- Anwendungsteile, die logisch zusammengehören, sollten Sie zu einem Lademodul vorbinden. Das können z.B. Programmteile sein, die sich gegenseitig referenzieren, oder die zu einer Folge von Teilprogrammläufen in einem Vorgang gehören.

- Alle Programmteile, die nachgeladen werden sollen, sollten Sie als LLM vorbinden. Der Start der Anwendung sowie das Laden von Lademodulen wird dadurch erheblich beschleunigt, da der dynamische Bindelader (DBL) LLMs wesentlich performanter verarbeiten kann als OMs.
- Die Anzahl der Lademodule, die zum Zeitpunkt des Starts des Anwendungsprogramms geladen werden (LOAD-MODE=STARTUP oder LOAD-MODE=(POOL,...,STARTUP), sollte so klein wie möglich gehalten werden. Aus diesem Grund sollten Sie nur die Teile der Anwendung mit STARTUP generieren, die häufig benötigt werden. Weil das Laden weniger großer Lademodule weniger Zeit benötigt als das Laden vieler kleiner Lademodule, sollten diese Teile des Anwendungsprogramms zu größeren Lademodulen vorgebunden werden.
- Es sollte nur **ein** anwendungslokaler Common Memory Pool (MPOOL ...,SCOPE=GROUP) definiert werden. In diesen können Sie mehrere vorgebundene Lademodule laden. Damit wird die Zeit zum Einrichten und Laden der Common Memory Pools verkürzt und somit die für den Start der Anwendung benötigte Zeit minimiert.
- OMs und LLMs sollten Sie in unterschiedlichen Programmbibliotheken bereitstellen, da auf diese Weise der Suchalgorithmus des BLS optimiert wird.
- Module, die weder Teilprogramme des Anwendungsprogramms noch Datenbereiche (AREA) sind (z.B. die Module der Laufzeitsysteme der Programmiersprache), brauchen Sie nicht einzeln mit dem Generierungstool KDCDEF als nachzuladende Module zu deklarieren, selbst wenn diese Module nicht statisch eingebunden sind. Sie können diese Module zu größeren Lademodulen (LLM) vorbinden und müssen nur den Namen des Lademoduls in der LOAD-MODULE-Anweisung generieren.

> **ACHTUNG!**
>
> Ob die mit der Anweisung LOAD-MODULE und den Operanden LOAD-MODULE in der PROGRAM- und AREA-Anweisung definierte Zuordnung der tatsächlichen Aufteilung der Lademodule in den Bibliotheken entspricht, kann von openUTM nicht verifiziert werden. openUTM verlässt sich beim Nachladen der Lademodule auf die in der UTM-Generierung gemachten Angaben. Sie müssen deshalb sicherstellen, dass die von Ihnen verwendeten Bindeprozeduren für die einzelnen Teile des Anwendungsprogramms mit den in der UTM-Generierung gemachten Angaben übereinstimmen. Andernfalls kann openUTM nicht sicherstellen, dass mit einem bestimmten Lademodul das gewünschte Programm in den Arbeitsspeicher geladen wird.

2.4 Regeln und Einschränkungen

Folgende Regeln und Einschränkungen müssen Sie bei der Aufteilung der Objekte in Lademodule sowie beim Binden und Nachladen der Objekte beachten:

- Die Event-Exits START, SHUT, INPUT und FORMAT und die Event-Services BADTAC, MSGTAC und SIGNON dürfen Sie keinem Lademodul zuordnen, das erst beim Aufruf eines Teilprogramms geladen wird, d.h. mit LOAD-MODE=ONCALL generiert ist. Der Grund dafür ist, dass die Event-Exits und -Services in jedem Prozess der Anwendung verfügbar sein müssen. Sind sie es nicht, dann bricht openUTM den Start des Prozesses ab.
  Die Lademodule, die die Event-Exits und -Services enthalten, sollten im Betrieb nicht ausgetauscht werden. Fehler beim Austausch der Lademodule können zu einem abnormalen Anwendungsende führen.
- Das Administrationsprogramm KDCADM dürfen Sie keinem Lademodul zuordnen, das mit LOAD-MODE=ONCALL generiert ist.
  Das Lademodul, in das das Administrationsprogramm KDCADM eingebunden ist, darf nicht ausgetauscht werden.
  Ist das Administrationsprogramm KDCADM beim Start der Anwendung nicht verfügbar, dann bricht openUTM den Start ab.
- Datenbereiche (AREAs) müssen entweder statisch zum Anwendungsprogramm gebunden werden, oder, falls möglich, als LOAD-MODULE mit LOAD-MODE(POOL, STARTUP) oder LOAD-MODE=STARTUP generiert werden.
- Das Start-LLM müssen Sie in einer Programmbibliothek bereitstellen. Es kann nicht in einer Datei stehen.
- Im Kommando START-EXECUTABLE-PROGRAM müssen Sie die folgenden Operanden angeben:

```
/     ,DBL-PARAMETERS=*PARAMETERS(                                  -
/            ,LOADING=*PARAMETERS(                                  -
/                           PROGRAM-MODE=*ANY                       -
/                          ,LOAD-INFORMATION=*REFERENCES)           -
...
/          ,ERROR-PROCESSING=*PARAMETERS(                           -
/                           UNRESOLVED-EXTRNS=*DELAY                -
/                          ,...))
```

  Die Angabe von PROGRAM-MODE=*ANY ist erforderlich, wenn die Anwendung in den oberen Adressraum geladen werden soll.
- Wenn Sie per Administration einen Programmaustausch anfordern, müssen Sie explizit die Version des neu zu ladenden Moduls angeben.
- Für Lademodule dürfen Sie nur die Namen von OMs oder LLMs angeben. Aus Performancegründen unterstützt openUTM nicht das Nachladen über CSECT- oder ENTRY-Namen.
- openUTM unterstützt maximal acht Common Memory Pools mit SCOPE=GROUP und acht Common Memory Pools mit SCOPE=GLOBAL.
- Zwei UTM-Anwendungen sollten möglichst unter unterschiedlichen BS2000-Benutzerkennungen gestartet werden, um Fehler zu vermeiden, die aufgrund von gleichen Modulnamen in shareable Teilen auftreten können. Module, die von mehreren Anwendungen benutzt werden, sollten daher in anwendungsglobale Common Memory Pools oder in nichtprivilegierte Subsysteme geladen werden.
- Durch einen Programmaustausch kann kein neu erzeugter Tabellenmodul aktiviert werden.
- Die Variantennummer des LMS ist beim Laden von Lademodulen mit BLS ohne Bedeutung.

- BLS unterstützt nur das Entladen von Programmteilen, die in **einem** Ladevorgang geladen wurden. Deshalb müssen Programmteile, die als separate Teile austauschbar sein sollen, in der Form bereitgestellt werden, in der sie geladen und ausgetauscht werden sollen. Soll ein Einzelmodul ausgetauscht werden, dann kann dieses Modul als OM oder LLM bereitgestellt werden; soll eine Gruppe von Lademodulen ausgetauscht werden, dann sind diese Lademodule als LLM vorzubinden.
- Lademodule, die TCB-Entries enthalten, dürfen im laufenden Betrieb nicht ausgetauscht werden.
- Module, die über die Autolink-Funktion nachgeladen wurden, können nicht ausgetauscht oder entladen werden. Ausnahme: Beim Austausch der gesamten Anwendung (z.B. mit KDCAPPL PROG=NEW) werden *alle* Anwendungsteile neu geladen.
- FHS kann nicht aus der TASKLIB nachgeladen werden. FHS-Format-Module können ausgetauscht werden.
- Beim Vorbinden der Lademodule zu LLMs ist zu beachten, dass die eingebundenen Elemente nicht in mehreren Lademodulen vorkommen sollten. Aus diesem Grund sind insbesondere beim Einbinden der Laufzeitsysteme der Programmiersprachen Einschränkungen zu beachten. Wie Sie die für das Anwendungsprogramm benötigten Module der Laufzeitsysteme einbinden können, ist im Abschnitt „Laufzeitsysteme binden“ beschrieben.

2.5 Shared Code nutzen

Viele Compiler bieten eine Option an, die es gestattet, bei der Übersetzung von Programmen einen shareable Teil zu erzeugen. Dieser muss nicht unbedingt in einem eigenen Objektmodul abgelegt werden, sondern kann zusammen mit dem nicht-shareable Teil in einem LLM stehen, der in einen Public und einen Private Slice unterteilt ist.

Folgende Objekte können shareable geladen werden:

- shareable Module der Teilprogramme
- Formate
- ein Teil der Module für die Formatierung (falls shareable)
- gemeinsam verwendbare Datenbereiche (Areas)
- benötigte Datenbank-Verbindungsmodule, falls diese shareable sind
- die Meldungsmodule
- Module der Laufzeitsysteme der Programmiersprachen, die shareable sind (siehe Handbücher der einzelnen Programmiersprachen)

Wenn Teile eines Teilprogramms shareable sein sollen, so müssen Sie das bereits beim Programmieren berücksichtigen. Was Sie beim Programmieren beachten müssen, ist abhängig vom Compiler, der das Teilprogramm übersetzt. Näheres dazu finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“ bzw. in der entsprechenden Sprachergänzung.

> ¡ Mit COBOL-Compiler ist entweder COBOL85 oder COBOL2000 gemeint.

COBOL-Teilprogramme, die mit dem COBOL85-Compiler übersetzt werden, sind shareable, wenn Sie bei der Übersetzung die Option `COMOPT GEN-SHARE-CODE=YES` angeben. Mit der Option `COMOPT GEN-LLM=YES` veranlassen Sie den Compiler, die Objekte in einen LLM mit Slices abzulegen.

COBOL-Teilprogramme, die mit dem COBOL2000-Compiler übersetzt werden, sind shareable, wenn Sie bei der Übersetzung folgende Optionen angeben:

```
COMPILER-ACTION=*MOD-GEN(SHAREABLE-CODE=*YES, MODULE-FORMAT=*LLM,...)
```

C/C++-Teilprogramme sind shareable, wenn Sie beim Übersetzen die folgende Option angeben:

```
COMPILER-ACTION=MODULE-GENERATION(SHAREABLE-CODE=YES,...)
```

Die entsprechende Option in weiteren Programmiersprachen finden Sie in den zugehörigen Benutzerhandbüchern.

Außerdem werden auch viele Laufzeitsysteme der Programmiersprachen mit einem shareable Teil ausgeliefert.

> ¡ Wenn Sie ein Teilprogramm austauschen möchten, das aus einem nicht-shareablen und einem shareablen Modul besteht, dann sollten Sie die shareable und nicht-shareable Teile zu einem LLM mit Slices vorbinden. Andernfalls können beim Austausch Konsistenzlücken entstehen. Konsistenzprobleme können Sie auch dadurch umgehen, indem Sie vor dem Austausch eines Lademoduls die TACs der in diesem Lademodul enthaltenen Teilprogramme sperren und diese erst nach dem Austausch der shareable und der nicht-shareable Teile wieder freigeben.

Shared Code laden

Die shareable Programmteile brauchen für alle Tasks der Anwendung(en) gemeinsam nur einmal geladen werden. In den task-lokalen Speicher müssen dann nur noch die nicht-shareable Teile geladen werden.

Sie haben verschiedene Möglichkeiten, shareable Objekte zu laden:

- in den **Systemspeicher** als nicht-privilegiertes Subsystem in den Klasse-3/4/5- oder 6-Speicher,
- in einen von openUTM verwalteten **Common Memory Pool** im Benutzerspeicher (Klasse-6-Speicher).

Sehen Sie dazu auch das Handbuch „BS2000/OSD Verwaltung von Subsystemen (DSSM/SSCM)“.

2.5.1 Shared Code im Systemspeicher

Shareable Teile der Anwendungsteilprogramme und Teile der Laufzeitsysteme können mit Hilfe der im BS2000 vorgesehenen Schnittstellen als shareable Programme in nichtprivilegierte Subsysteme geladen werden.

Shared Code, der in nicht-privilegierte Subsysteme geladen ist, kann während des Betriebs der Anwendung ausgetauscht werden.

Die **shareable** Module müssen durch den Administrator in den Speicher geladen werden, bevor die Anwendung gestartet wird. Sehen Sie hierzu auch die Hinweise am Ende dieses Abschnitts. Im Gegensatz zu Shared Code in Common Memory Pools, der unter Kontrolle von openUTM geladen und ausgetauscht werden kann, können Sie die shareable Module im Systemspeicher nicht selbst verwalten.

Nicht-shareable Teile der Teilprogramme müssen mit der PROGRAM-Anweisung beschrieben und ggf. mit der LOAD-MODULE-Anweisung einem Lademodul zugeordnet werden.
Das Lademodul bzw. dessen private Slice wird beim Start des Anwendungsprogramms dynamisch in den task-lokalen Speicher (Klasse-6-Speicher) geladen. Über die Externbezüge zu den shareable Modulen werden die Verknüpfungen in den Shared Code dynamisch hergestellt.
Die Lademodule (OM-Format), die die shareable Module des Teilprogramms enthalten, müssen nicht zusammen mit den Lademodulen, die die nicht-shareable Programmteile enthalten, in einer Programmbibliothek stehen.

Hinweise

- Subsysteme sollten im LLM-Format erstellt werden, da diese Subsysteme im laufenden Betrieb der Anwendung **ohne** Konsistenzlücke ausgetauscht werden können. Beim Austausch des Subsystems wird eventuell der Public Slice des alten LLMs in den tasklokalen Teil des Anwendungsprogramms geladen, bis auch der Private Slice ausgetauscht wird (z.B. mit KDCPROG initiiert).

  Subsysteme im OM-Format haben hingegen beim Austausch im laufenden Betrieb eine Konsistenzlücke, die zwischen dem DSSM-Subsystemkommando

  ```
  START-SUBSYSTEM SUBSYSTEM-NAME=subsystemname -
                , VERSION=new-version         -
                , VERSION-PARALLELISM=*EXCHANGE-MODE
  ```

  und dem UTM-Administrationskommando (oder einen entsprechenden KDCADMI-Aufruf an der Programmschnittstelle zur Administration)

  ```
  KDCPROG LOAD-MODULE=load-module,VERSION=new-version
  ```

  existiert. Hier kann es beispielsweise vorkommen, dass nach dem DSSM-Kommando und vor dem UTM-Administrationskommando ein UTM-Anwendungsprogramm mit PEND ER beendet wird und die betroffene Task des UTM-Anwendungsprogramms mit dem alten Private Slice und einem neuen Public Slice in einem neuen Subsystem startet.
- Beim Austausch von Subsystemen verschiedener Versionen, deren Module in der gleichen Bibliothek stehen (Bedingung zum Austausch von Lademodulen bei openUTM), ist darauf zu achten, dass unterschiedliche LINK-ENTRYs angegeben werden, da sonst das DSSM ggf. den Austausch nicht durchführen kann.
- Beim Erstellen eines Subsystems sollte das entsprechende LLM-Format (z.B. Format 2) verwendet werden, da hier nur ein Subsystemeintrag angegeben werden muss, siehe auch „BINDER Benutzerhandbuch“.

2.5.2 Shared Code in Common Memory Pools

Ein Common Memory Pool ist ein Bereich im Benutzerspeicher (Klasse-6-Speicher). In einen Common Memory Pool können Sie shareable Objekte laden, die beim Binden des Anwendungsprogramms nicht statisch dazugebunden wurden. openUTM bietet zwei Möglichkeiten, einen Common Memory Pool einzurichten:

- als anwendungslokalen Pool

  Auf diesen Pool können alle Teilprogramme einer Anwendung zugreifen. Es können auch Teilprogramme anderer Anwendungen auf den Pool zugreifen, sofern diese Anwendungen unter derselben BS2000-Benutzerkennung gestartet wurden.
- als anwendungsglobalen Pool

  Teilprogramme **mehrerer** Anwendungen können auf den gleichen Common Memory Pool zugreifen, unabhängig von den Benutzerkennungen, unter denen diese Anwendungen gestartet wurden.

Jeden Common Memory Pool müssen Sie mit der KDCDEF-Anweisung MPOOL generieren. In der MPOOL-Anweisung legen Sie fest, ob ein Common Memory Pool anwendungsglobal oder anwendungslokal eingerichtet wird.

2.5.2.1 Anwendungslokaler Pool

Ein anwendungslokaler Pool wird von der ersten Task einer Anwendung angelegt. Alle weiteren Tasks dieser Anwendung greifen ebenfalls darauf zu, d.h. der Pool wird im Sinne der Memory-Pool-Verwaltung des BS2000 mit SCOPE=GROUP angelegt. Auch Tasks weiterer UTM-Anwendungen, die unter derselben Benutzerkennung gestartet wurden, können sich an diesen Common Memory Pool anschließen.

Der anwendungslokale Pool bietet folgende Vorteile für den Anwender:

- Die Nutzung der in diesen Pool geladenen Programme ist nur von dieser Benutzerkennung aus möglich.
- Der Benutzer hat die volle Kontrolle über die Verwendung von shareable Objekten, unabhängig von der System-Initialisierung.
- Der Pool existiert nur so lange die Anwendung läuft, bzw. wenn mehrere Anwendungen auf den Pool zugreifen, bis die letzte Anwendung beendet wird. Mit dem Ende der Anwendung wird der Pool aufgelöst und die Programme werden - wie alle anderen Anwenderprogramme auch - entladen.
- Die in einen Common Memory Pool geladenen Programme können unter bestimmten Voraussetzungen während eines Anwendungslaufs dynamisch ausgetauscht werden. Näheres hierzu finden Sie in Kapitel „Programmaustausch im Betrieb“.
- Der physikalische Speicher wird besser ausgenutzt.

Mit dem Beenden der (letzten) Anwendung wird der Pool aufgelöst und die Programme werden - wie alle anderen Anwenderprogramme auch - entladen.

> **ACHTUNG!**
>
> Alle Module, Formate und Datenbereiche haben innerhalb eines Pools die gleiche Zugriffsberechtigung.
>
> Programme in Common Memory Pools dürfen keine offenen Externverweise auf Adressen außerhalb des Common Memory Pools enthalten, die auf Module in anderen Common Memory Pools oder nicht-privilegierten Subsystemen im Klasse 5/6-Speicher zeigen. Diese CSECTs und ENTRYs werden nicht gefunden. Weiterhin dürfen offene Externverweise nur auf Shared Code zeigen, der für alle Tasks auf derselben Adresse liegt. Sehen Sie dazu die DSSM-Handbücher.

2.5.2.2 Anwendungsglobaler Pool

Ein anwendungsglobaler Pool wird für Module verwendet, die in mehreren Anwendungen gebraucht werden wie beispielsweise Formate, Verbindungsmodul für Formatierung, Datenbank-Verbindungsmodule, Sprach-Laufzeitmodule, gemeinsam nutzbare Datenbereiche, aber auch Routinen in Teilprogrammen und der Meldungsmodul.

Ein anwendungsglobaler Pool wird von der ersten Task der ersten UTM-Anwendung eingerichtet. An diesen Pool können sich alle weiteren Tasks derselben und anderer Anwendungen anschließen, d.h. der Pool wird im Sinne der Memory-Pool-Verwaltung des BS2000-Systems mit SCOPE=GLOBAL angelegt.

Ein anwendungsglobaler Pool bietet ähnliche Vorteile wie ein anwendungslokaler Pool. Er hat jedoch den Nachteil, dass die in einen solchen Pool geladenen Programme während eines Anwendungslaufs nicht dynamisch ausgetauscht werden können.

Werden globale Common Memory Pools gleichen Inhalts/Namens in mehreren UTM-Anwendungen verwendet, muss in der KDCDEF-Anweisung MPOOL der Parameter PAGE=X'xxxxxxxx' mit gleicher Adresse in allen Anwendungen angegeben werden. Die mit PAGE= angegebene Adresse ist so zu wählen, dass der damit reservierte Adressbereich in all diesen Anwendungen verfügbar ist.

Beispiel

Anwendung 1 verwendet den globalen Pool MPEINS. Anwendung 2 verwendet ebenfalls diesen Pool und zusätzlich den Pool MPZWEI. Folgende KDCDEF-Steueranweisungen werden benötigt:

UTM-Generierung Anwendung 1:

```
MPOOL MPEINS, SCOPE=GLOBAL, PAGE=X'01000000', SIZE=...
```

UTM-Generierung Anwendung 2:

```
MPOOL MPZWEI, SCOPE=GLOBAL, SIZE=...

MPOOL MPEINS, SCOPE=GLOBAL, PAGE=X'01000000', SIZE=...
```

Eine Alternative für die Verwendung von PAGE= ist, die gemeinsamen Pools in allen Anwendungen in der gleichen Reihenfolge zu generieren. Diese MPOOL-Anweisungen müssen außerdem als erste MPOOL-Anweisungen angegeben werden.

2.5.2.3 Shareable Objekte generieren, die in einen Common Memory Pool geladen werden

Sollen Teile Ihres Anwendungsprogramms in einen Common Memory Pool geladen werden, dann beachten Sie bitte Folgendes:

- Aus Performancegründen sollten Sie, soweit möglich, alle shareable Teile eines Anwendungsprogramms, die in einen Common Memory Pool geladen werden sollen, zu **einem** Lademodul zusammenfassen.
- Das vom Compiler erzeugte shareable Code-Modul des Programms muss in einem LLM oder OM enthalten sein.
- Erzeugt ein Compiler für den shareable und den nicht-shareable Teil zwei getrennte Objektmodule, dann sollten Sie diese Module mit dem Binder zu einem LLM mit Slices vorbinden.
- LLMs mit Slices können Sie mit einer einzigen LOAD-MODULE-Anweisung generieren:

```
LOAD-MODULE llm-name ,VERSION=version         -
  ,LOAD-MODE=(POOL,poolname,{STARTUP|ONCALL}) -
  ,LIB=program-lib                            -
  ,ALTERNATE-LIBRARIES={YES|NO}
```

  Durch diese Anweisung wird der Public Slice des LLM in den Common Memory Pool *poolname* geladen, der Private Slice wird entweder bei Anwendungsstart (STARTUP) oder bei Programmaufruf (ONCALL) nachgeladen. Für die durch openUTM aufzurufenden Programme dieses LLMs sind zusätzlich PROGRAM-Anweisungen notwendig.

  > Alternativ dazu können Sie auch das shareable und das nicht-shareable Modul durch zwei LOAD-MODULE-Anweisungen generieren. Dies sollten Sie jedoch möglichst vermeiden, da sich diese beiden Module nicht ohne Konsistenzlücke austauschen lassen.

- Einen gemeinsam nutzbaren Datenbereich (Area), der in den Common Memory Pool geladen werden soll, müssen Sie mit einer AREA-Anweisung beschreiben. Die Area muss dann in dem Lademodul enthalten sein, der folgendermaßen generiert wird:

```
LOAD-MODULE ar-share  ,VERSION=version        -
  ,LOAD-MODE=(POOL,poolname,NO-PRIVATE-SLICE) -
  ,LIB=libname
```

  AREAs, denen beim Übersetzen oder durch den Binder das Attribut PUBLIC zugeordnet wurde, können auch zusammen mit anderen Modulen in einen LLM mit Slices vorgebunden werden. Dieses LLM kann dann folgendermaßen generiert werden:

```
LOAD-MODULE llm-with-slices ,VERSION=version  -
  ,LOAD-MODE=(POOL,poolname,{STARTUP|ONCALL}),LIB=libname
```

Beispiel

Das Beispiel geht davon aus, dass zum Übersetzen ein COBOL-Compiler verwendet wurde und dass der Compiler die Objekte in ein LLM abgelegt hat. In den anwendungslokalen Pool LCPOOL sollen die shareable Module der COBOL-Teilprogramme TP1 und TP2, die Formatbeschreibungen FORMAT1 und FORMAT2 und der Datenmodul DATAMOD geladen werden. LCPOOL soll auf Adresse X’020000’ geladen werden, 128 KB belegen können und schreibgeschützt sein.

```
/MPOOL        LCPOOL,SIZE=2,SCOPE=GROUP,ACCESS=READ,PAGE=X'20000'
/LOAD-MODULE  LLM-LCPOOL,VERSION=1,            -
/             LOAD-MODE=(POOL,LCPOOL,STARTUP), -
/             LIB=libname
/PROGRAM      TP1 ,LOAD-MODULE=LLM-LCPOOL,COMP=ILCS
/PROGRAM      TP2 ,LOAD-MODULE=LLM-LCPOOL,COMP=ILCS
/AREA         DATAMOD,LOAD-MODULE=LLM-LCPOOL
```

Die Objektmodule müssen Sie vor dem Start der Anwendung zu dem LLM LLM-LCPOOL vorbinden und dabei in der BINDER-Anweisung START-LLM-CREATION die Option BY-ATTRIBUTES(PUBLIC=YES) angeben, wodurch das LLM in eine Public und einen Private Slice unterteilt wird. Das so erzeugte LLM müssen Sie in der Bibliothek *libname* bereitstellen.

3 Anwendungsprogramm erzeugen

Die Teilprogramme implementieren die Anwendungslogik und müssen vor dem Start der Anwendung geschrieben und übersetzt werden. Sehen Sie dazu das openUTM-Handbuch „Anwendungen programmieren mit KDCS“.

Damit die Teilprogramme unter openUTM ablaufen können, wird das UTM-Anwendungsprogramm wie folgt erzeugt:

- ROOT-Tabellen-Source, die Sie mit dem Generierungstool KDCDEF erzeugt haben, assemblieren.
- ROOT-Systemmodule und die Sprach-spezifischen Laufzeitsysteme zu einem Start-LLM binden. Es können weitere Anwendungs-spezifische Teile wie Meldungsmodule, Format-Bibliotheken und Teilprogramme in das Start-LLM eingebunden werden.

Das ROOT-Tabellenmodul kann beim Start der Anwendung dynamisch nachgeladen werden.

Die einzelnen Schritte, welche zum Erzeugen eines UTM-Anwendungsprogramms notwendig sind, zeigt die folgende Abbildung.

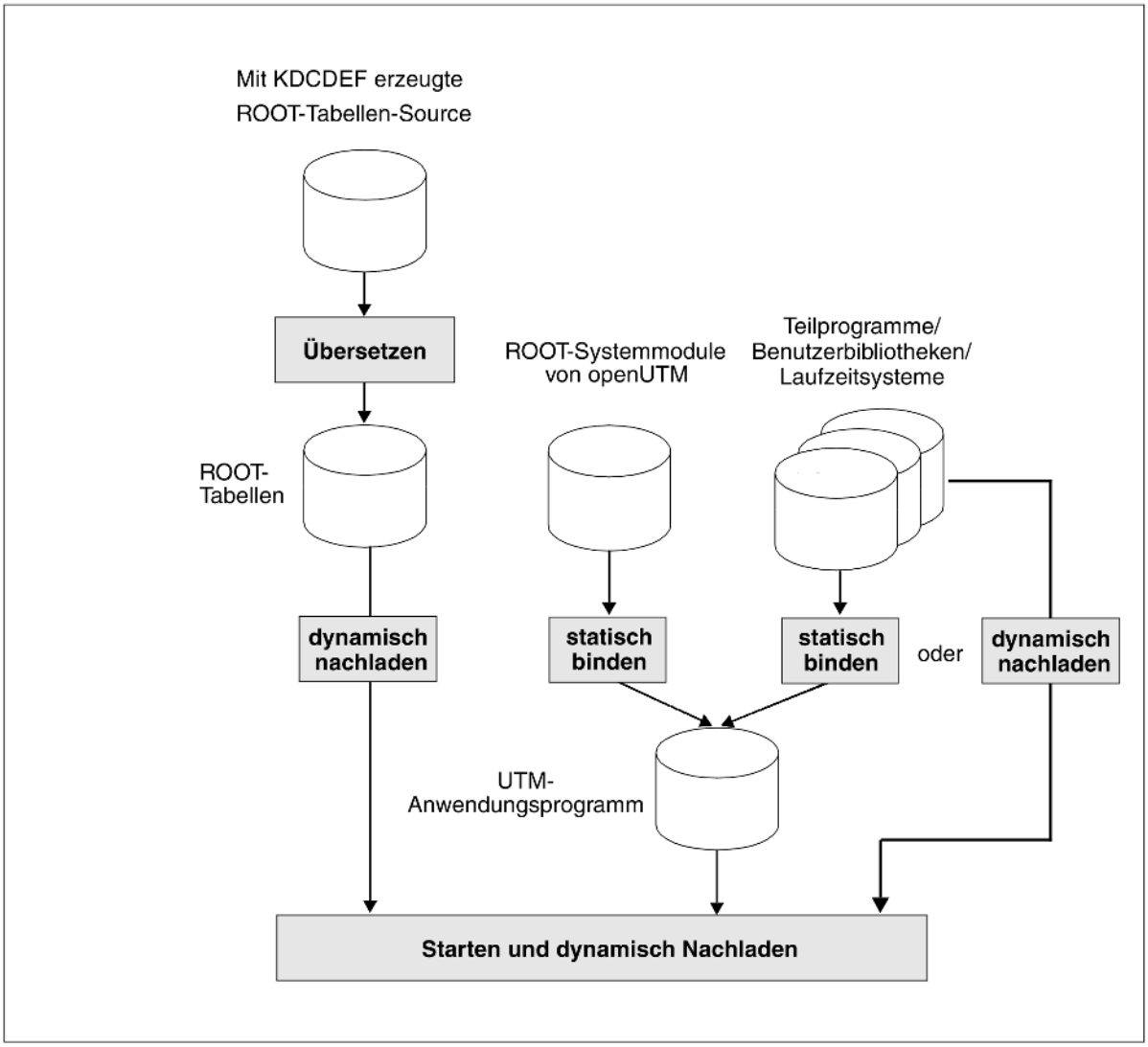


Bild 2: Übersicht: Erzeugen und Starten des UTM-Anwendungsprogramms mit dynamischem Nachladen

Die ROOT-Tabellen können wie andere Anwendungsteile auch statisch zum Anwendungsprogramm gebunden werden. Sehen Sie dazu die folgende Abbildung.

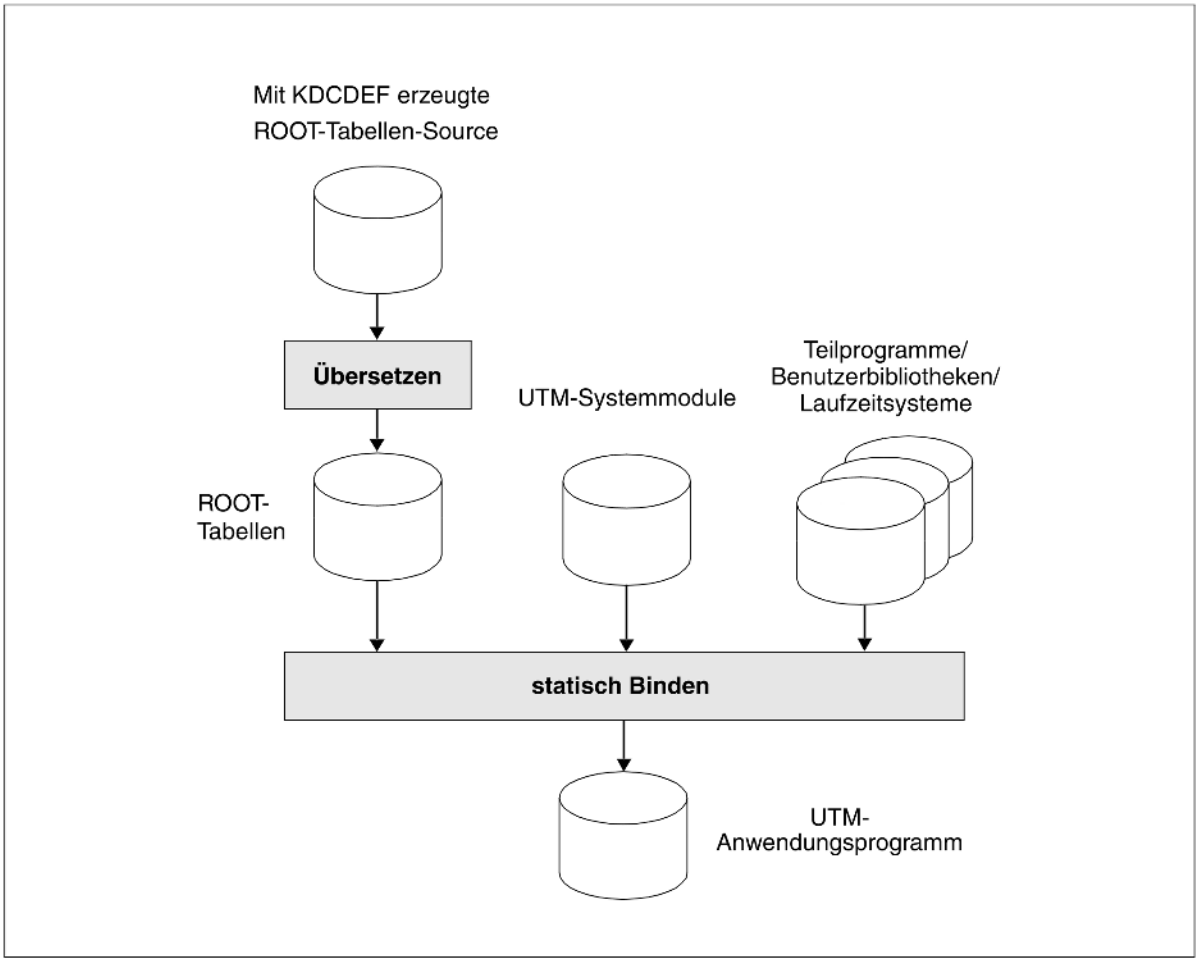


Bild 3: Übersicht: Erzeugen des UTM-Anwendungsprogramms ohne dynamisches Nachladen

Main Routine KDCROOT

Aus den ROOT-Systemmodulen, die mit openUTM ausgeliefert werden, entsteht beim Binden die Main Routine KDCROOT als Teil des Anwendungsprogramms. KDCROOT fungiert beim Ablauf der Anwendung als steuerndes Hauptprogramm und übernimmt u.a. folgende Aufgaben:

- Verbindung zwischen den Teilprogrammen und den UTM-Systemfunktionen
- Ablauf-Koordination von Teilprogrammen unterschiedlicher Programmiersprachen
- Kopplung zu Datenbanken
- Zusammenarbeit mit Formatierungssystemen

Weiter enthält KDCROOT Bereiche für variable Daten sowie die Nachrichtenbereiche. Nähere Informationen zur Main Routine KDCROOT sind dem openUTM-Handbuch „Anwendungen programmieren mit KDCS“ zu entnehmen.

Der Standardname für die ROOT-Tabellen-Source ist ROOT.SRC.ASSEMB.*rootname*. Die von KDCDEF erzeugte Source für die ROOT-Tabellen müssen Sie assemblieren, beispielsweise mit ASSEMBH-GEN. Die ROOT-Systemmodule, die zum Anwendungsprogramm dazugebunden werden, sind in der Modulbibliothek SYSLNK.UTM.070 enthalten.

3.1 Bestandteile des Anwendungsprogramms

Ein UTM-Anwendungsprogramm besteht aus einer Reihe von Modulen, die spätestens zur Ablaufzeit als ein Programm gebunden sein müssen.

Die folgenden Module sind zum Ablauf notwendig:

- rootmodul

  Übersetztes ROOT-Tabellenmodul, das Sie in einer Programmbibliothek oder Bindemodulbibliothek ablegen müssen. Wenn Sie die ROOT-Nachladetechnik verwenden, geben Sie diese Bibliothek beim Startparameter TABLIB= und den Namen des Moduls beim Startparameter ROOTNAME= an.

- ROOT-Systemmodule

  Module, die die UTM-Anwendung zum Ablauf benötigt. Sie stehen in der Bibliothek SYSLNK.UTM.070.

- Teilprogramme

  Vom Anwender erstellte Teilprogramme der Anwendung, die Sie übersetzen und in eine oder mehrere Programmbibliotheken oder Bindemodulbibliotheken eintragen lassen müssen.

- Datenbank-Verbindungsmodule

  Dienen zum Anschluss der UTM-Anwendung an die Datenbanksysteme, die bei der KDCDEF-Generierung im Operanden TYPE der DATABASE-Anweisungen angegeben wurden. Wie die DB-System diese Module bereitstellen, ist in den Handbüchern der entsprechenden DB-Systeme beschrieben.

- Verbindungsmodul für das Formatierungssystem

  Dient zum Anschluss der UTM-Anwendung an ein Formatierungssystem, das im Operanden TYPE der FORMSYS-Anweisung generiert wurde. Wie das Formatierungssystem dieses Modul bereitstellt, ist in den Handbüchern der entsprechenden Formatierungssysteme beschrieben.

- Administrations- und UTM-SAT-Administrationsteilprogramm

  Programme zur Administration der UTM-Anwendung. Sie können ein solches Programm unter Verwendung der Programmschnittstelle zur Administration entweder selbst schreiben oder die mit openUTM ausgelieferten

  Standard-Administrationsprogramme KDCADM und KDCSADM verwenden. Jedes Administrationsprogramm müssen Sie in einer eigenen PROGRAM-Anweisung generieren. Die Lademodule der ausgelieferten Administrationsprogramme stehen in der Bibliothek SYSLNK.UTM.070 zur Verfügung.
  Das Administrationsprogramm KDCADM wird immer benötigt, auch dann, wenn Sie ein selbst erstelltes Administrationsprogramm verwenden.

- Laufzeitsystem für den KDCROOT

  wird immer benötigt und in der Bibliothek SYSLNK.UTM.070.SPLRTS ausgeliefert. Zusätzlich braucht der KDCROOT eine der Bibliotheken SYSLNK.CRTE bzw. SYSLNK.CRTE.PARTIAL-BIND, wobei die zweite Bibliothek Performancevorteile bringt und daher vorzuziehen ist. CRTE ist Software-Voraussetzung für openUTM und enthält das C-Laufzeitsystem, das ILCS und das COBOL-Laufzeitsystem. Das C-Laufzeitsystem und das ILCS werden von openUTM immer benötigt.

- Laufzeitsysteme der Programmiersprachen

  werden benötigt, wenn mindestens ein Teilprogramm der Anwendung in einer höheren Programmiersprache geschrieben wurde. Falls die Anwendung ILCS-fähige Teilprogramme enthält, müssen Sie darauf achten, dass die höchste zur Verfügung stehende ILCS-Version eingebunden wird. Weitere Bindehinweise dazu erhalten Sie in den Informationen zu ILCS im Benutzerhandbuch CRTE. Die ILCS-Module sind in der CRTE-Bibliothek enthalten.

  Welche Laufzeitsysteme mit welchen Compilerversionen verwendet werden dürfen, ist in Abschnitt „Compiler-Versionen, Laufzeitsysteme, KDCDEF-Optionen“ beschrieben.

- Sprachanschlussmodule

  werden für jede generierte Programmiersprache oder deren Linkage benötigt. Für SPL, Assembler, COB1 und ILCS-Linkage sind diese Anschlussmodule schon in das Laufzeitsystem für KDCROOT eingebunden. Für alle anderen Programmiersprachen müssen Sie die Anschlussmodule spätestens zum Anwendungsstart aus Sprach-spezifischen Bibliotheken einbinden.
- gemeinsam verwendbare Bereiche

  (siehe AREA-Anweisung von KDCDEF)
- benutzereigene Meldungsmodule

  falls benutzereigene Meldungsmodule statisch eingebunden werden sollen.

3.2 Binden des Anwendungsprogramms

UTM-Anwendungen nutzen die Funktionalität des BLS zum Laden von LLM (Link and Load Module, Bindelademodule) und von OM (Object Module, Bindemodul). Damit bieten sich erweiterte Möglichkeiten:

- bei der Verwaltung der Objekte durch die Nutzung von LLMs, die in eine Private und eine Public Slice unterteilt sind.

  Dadurch können shareable und nicht-shareable Teile eines Objekts, die in einem Compilerlauf erzeugt wurden, in einem Container verwaltet werden. Dies vermeidet weitgehend Inkonsistenzen beim Vorbinden und Austauschen von Objekten.
- beim Laden eines Objekts (Programm, Area).

  Ein Objekt kann statisch eingebunden oder beim Anwendungsstart nachgeladen werden. Für Teilprogramme besteht darüberhinaus die Möglichkeit, sie erst zum Zeitpunkt des ersten Aufrufs nachzuladen.
- beim Programmaustausch.

  Es kann ein einzelnes Objekt (LLM oder OM) oder eine vorgebundene Gruppe von Objekten (LLM oder OM) ausgetauscht werden.

Es braucht nur ein Teil der Objekte der Anwendung statisch gebunden werden. Dieser Teil des Anwendungsprogramms heißt **Start-LLM** und wird mit dem Kommando START-EXECUTABLE-PROGRAM FROM-FILE=*LIBRARY(...) geladen und gestartet. In das Start-LLM müssen Sie die ROOT-Systemmodule und die Objekte (Teilprogramme, Module und Datenbereiche) einbinden, die keinem Lademodul oder einem Lademodul mit LOAD-MODE=STATIC zugeordnet sind.

Das vom KDCDEF erzeugte ROOT-Tabellenmodul muss nicht statisch gebunden werden, d.h. es kann beim Start der Anwendung dynamisch nachgeladen werden.

Die anderen Teile des Anwendungsprogramms können in Form von Lademodulen bereitgestellt werden. Ein Lademodul ist

- entweder ein Bindelademodul, genannt Link Load Module (=**LLM**), das als Element vom Typ L in einer Programmbibliothek steht,
- oder ein Objektmodul (=**OM**), das als Element vom Typ R in einer Bindemodulbibliothek steht.

Aus Gründen der Performance sollten Sie jedoch alle Lademodule mit dem BINDER zu LLM vorbinden und als Elemente vom Typ L in einer Programmbibliothek bereitstellen.

Bei der UTM-Generierung müssen Sie für jedes Bindelademodul und jedes Objektmodul, das Sie nicht statisch in das START-LLM einbinden, genau eine LOAD-MODULE-Anweisung eingeben und dabei angeben, wann das Modul dazugebunden und wohin es geladen werden soll. Siehe dazu openUTM-Handbuch „Anwendungen generieren“.

Existieren beim Binden der Anwendung unbefriedigte Externverweise auf Teilprogramme, lässt sich die Anwendung trotzdem starten. Können diese Externverweise nicht durch dynamisches Nachladen aufgelöst werden, sind die zugeordneten Transaktionscodes für die Anwendung ungültig, d.h. sie können nicht benutzt werden.

3.2.1 LLMs mit Slices

Der BINDER bietet Ihnen die Möglichkeit, CSECTs, die in einem LLM enthalten sind, nach bestimmten Kriterien zu einer ladbaren Einheit zusammenzufassen, die **Slice** genannt wird. Dabei wird zwischen Slices unterschieden, die nach Attributen von CSECTs gebildet werden (Slices by Attributes) und solchen, die vom Benutzer definiert werden (User defined Slices).

openUTM kann nur LLMs mit Slices by Attributes verarbeiten und unterstützt dabei das Attribut PUBLIC. Viele Compiler bieten die Funktion an, die erzeugten Objekte in einem LLM mit Public und Private Slice abzulegen. Die Public Slice enthält dann den shareable Teil des Objekts und die Private Slice den Teil, der nicht-shareable ist. Dieses Verfahren vereinfacht die Verwaltung von shareable und nicht-shareable Teilen eines Programms, da sich beide in einem LLM befinden, siehe auch Abschnitt „LLMs mit Public/Private Slice binden“. Wenn Sie ein solches LLM bei openUTM entsprechend generieren, dann lädt openUTM die Public Slice in einen Common Memory Pool und die Private Slice in den task-lokalen Speicher.

Wenn beim Laden oder beim Austausch die Public Slice eines LLM nicht in den Common Memory Pool geladen werden kann, wird dieser Task nach einer entsprechenden K078-Meldung beendet und der Anwendungstausch abgebrochen.

3.2.2 Binden von LLMs

Als LLMs sollten Sie alle Modulgruppen vorbinden, die in einem logischen Zusammenhang stehen. Dies können sein:

- Teile der Laufzeitsysteme
- logisch zusammengehörige Teile einer UTM-Anwendung, wie z.B. alle Teilprogramme eines oder mehrerer Vorgänge einschließlich Vorgangsexits
- FHS-Formate samt der zugehörigen Teilprogramme

Damit beim dynamischen Binden eines LLM die Externverweise dieses LLM befriedigt werden können, müssen Sie beim Erzeugen des LLM unbedingt folgende BINDER-Anweisungen angeben:

```
/START-LLM-CREATION    INTERNAL-NAME=int-name     - ———————————————————  (1)
/                     ,INTERNAL-VERSION=int-vers  -
/                     ,...
/SET-EXTERN-RESOLUTION SYMBOL-TYPE=*REFERENCES    - ———————————————————  (2)
/                     ,RESOLUTION=*STD
/SAVE-LLM           ...,ELEMENT=*INTERNAL-NAME(    - ———————————————————  (3)
/                       VERSION=*INTERNAL-VERSION) -
/                     , ...
```

1. Hier legen Sie schon Name und Version des LLM fest, unter denen das LLM zum Abschluss des Bindens in der Bibliothek abgespeichert wird.
2. Die Behandlung unbefriedigter Externverweise wird festgelegt. Die angegebenen Operandenwerte sind die Standardwerte, die auf keinen Fall geändert werden dürfen.
3. Das LLM muss unter dem internen Namen abgespeichert werden; das Format des LLM muss korrekt sein (mindestens LLM-Format 2 wie im Beispiel).

   Wird beim Abspeichern ein anderer Name als der interne gewählt und wird bei der UTM-Generierung LOAD-MODULE,...,ALTLIB=YES angegeben, so kann openUTM bei einem Programmaustausch das Modul nicht finden und meldet einen UNBIND-Fehler (`K078 UNBIND 0C010174`).

Mehrfach vorkommende Elemente

Beim Vorbinden von LLMs müssen Sie beachten, dass eingebundene Elemente
(Sub-LLM / OM) nicht in mehreren, evtl. vorgebundenen LLM / OM vorkommen. Falls dies doch der Fall ist, müssen Sie die externen Namen, die in mehrfach eingebundenen Modulen enthalten sind, durch die BINDER-Anweisung MODIFY-SYMBOL-VISIBILITY maskieren.

Autolink-Funktion für das Nachladen von Lademodulen

Die Autolink-Funktion wird in der LOAD-MODULE-Anweisung durch den Operanden ALTERNATE-LIBRARIES gesteuert.

- ALTERNATE-LIBRARIES=NO
  schaltet die Autolink-Funktion des BLS aus. D.h. für alle dynamisch generierten Lademodule, die auf diese Weise geladen werden, müssen alle offenen Verweise eines solchen Lademoduls durch die zum Ladezeitpunkt geladenen Module (Start-LLM und andere Lademodule) und durch den Shared Code befriedigt werden. Externverweise ins Laufzeitsystem werden bei Lademodulen, die nur aus C- oder Datenobjekten bestehen, immer dann befriedigt, wenn beim Binden des Start-LLM ein RESOLVE-BY-AUTOLINK auf die Bibliothek SYSLNK.CRTE.PARTIAL-BIND angegeben wurde.

- ALTERNATE-LIBRARIES=YES
  schaltet die Autolink-Funktion des BLS ein. Diese können Sie für solche Lademodule verwenden, die beim Laden und Austausch weitere Module des Laufzeitsystems benötigen, die sich noch nicht im Speicher befinden. Beim Binden werden offene Externverweise zunächst im Link-Kontext und dann im Shared Code gesucht (siehe auch Handbuch „Bindelader - Starter“). Wenn danach Externverweise unbefriedigt geblieben sind, wird die Bibliothek durchsucht, die Sie im LIB-Operanden angegeben haben. Wird eine passende Definition gefunden, dann wird das zugehörige Modul eingebunden und die Suche abgebrochen. Ansonsten wird in den Bibliotheken weitergesucht, die Sie vor dem Start der Anwendung durch ein SET-FILE-LINK-Kommando mit den Linknamen BLSLIB*nn* (wobei 00<=*nn*<=99), verknüpft haben. Die Bibliotheken werden in aufsteigender Reihenfolge der nn bearbeitet.

  Die Behandlung der Module des Laufzeitsystems in Verbindung mit UTM-Anwendungen wird ausführlich auf "Laufzeitsysteme binden" beschrieben.

Der Autolink-Mechanismus sollte nicht auf benutzereigene Bibliotheken angewandt werden, da bei einem Programmaustausch nur das Lademodul und nicht die ganze Ladeeinheit, die zusätzlich alle durch Autolink geladenen Module enthält, entladen wird.

Die Autolink-Funktion für das Start-LLM kann über die Parameter des Kommandos START-EXECUTABLE-PROGAM beeinflusst werden, siehe auch das Beispiel im Abschnitt "Module laden".

Beispiel

- UTM-Generierung und LINK-Anweisung

  UTM-Generierung:

```
LOAD-MODULE     lm-name -
                ,VERSION             = llm-version -
                ,LIB                 = lm-lib      -
                ,LOAD-MODE           = ONCALL      -
                ,ALTERNATE-LIBRARIES = YES
```

  LINK-Anweisung vor Start der Anwendung:

```
 /SET-FILE-LINK  LINK-NAME = BLSLIB00,FILE-NAME = $userid.SYSLNK.CRTE.PARTIAL-BIND
```

  Zunächst wird mit der LOAD-MODULE-Anweisung das Lademodul *lm-name* mit LOAD-MODE=ONCALL generiert, d.h. *lm-name* wird erst beim Aufruf eines darin enthaltenen Programms nachgeladen. Im Beispiel sollte *lm-name* COBOL-Objekte enthalten.

  Um offene Externverweise beim Laden zu befriedigen, wird erst die im LIB-Operanden angegebene Bibliothek *lm-lib* durchsucht und dann die Bibliothek $*userid*.SYSLNK.CRTE.PARTIAL-BIND, die über das SET-FILE-LINK-Kommando zugewiesen ist.

- BINDER-Anweisungen

```
/START-BINDER
//START-LLM-CREATION     INTERNAL-NAME    = llm-name   -
//                      ,INTERNAL-VERSION = llm-version
//INCLUDE-MODULES        LIBRARY=benutzerbibliothek    -
//                      ,ELEMENT=teilprogrammname
//RESOLVE-BY-AUTOLINK    LIBRARY=(bibliothek1,bibliothek2..)
//SET-EXTERN-RESOLUTION  SYMBOL-TYPE=*REFERENCES       -
//                      ,RESOLUTION=*STD
//SAVE-LLM               LIBRARY=llm-bibliothek        -
//                      ,ELEMENT=*INTERNAL-NAME(       -
//                       VERSION=*INTERNAL-VERSION)    -
//                      ,FOR-BS2000-VERSIONS=*FROM-OSD-V4(...)
//END
```

In diesem BINDER-Lauf wird ein Teilprogramm der Anwendung zu einem Lademodul gebunden. KDCROOT wird hier nicht mit eingebunden, weswegen u.a. der Entry KDCS unbefriedigt bleibt.

3.2.3 LLMs mit Public/Private Slice binden

Wenn Sie mit dem Binder shareable Module im OM-Format zu LLMs mit einer Public Slice binden, dann müssen Sie darauf achten, dass die CSECTs dieser Module das Attribut PUBLIC besitzen. Ist dies nicht der Fall, legt der Binder diese CSECTs in die Private Slice.

Auftreten kann dies zum Beispiel bei AREAs, die vorher in shareable Modulen im OM-Format gewesen sind (hier hatte das Attribut keine Bedeutung), oder bei allen shareable COBOL-Objekten. Sie können die Objekte anpassen, indem Sie

- bei AREAs das Attribut PUBLIC hinzufügen und dann neu übersetzen,
- bei shareable COBOL-Objekten die zugehörigen Sourcen neu übersetzen und dabei im LLM-Format abspeichern
- oder das PUBLIC-Attribut mit dem Binder setzen.

Das folgende Beispiel zeigt die dritte Möglichkeit.

Beispiel

Ein Lademodul soll als LLM vorgebunden werden. Es soll bestehen aus

- den COBOL-Objekten AFPUT und COBECHO mit den nicht-shareable Teilen AFPUT und COBECHO und den shareable Teilen AFPUT@ und COBECHO@
- sowie den shareable AREAs AREA1, AREA2 und AREA3, denen allen das PUBLIC-Attribut fehlt.

Dann ergeben sich folgende Binder-Anweisungen:

```
//START-LLM-CREATION                                                -
//INTERNAL-NAME  = EXAMPLE-LLM                                      -
//                ,INTERNAL-VERSION    = 001                        -
//                ,SLICE-DEFINITION    = BY-ATTR( PUBLIC = YES )
//REMARK ------------------------------------------------------------
//REMARK ------------------------------------------------------------
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                -
//          ,ELE = COBECHO
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                -
//          ,ELE = AFPUT
//REMARK ------------------------------------------------------------
//REMARK ------------------------------------------------------------
//BEGIN-SUB-LLM SUB-LLM-NAME=OM-WITHOUT-PUBLIC-ATTRIBUTE
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                -
//          ,ELE = AFPUT@
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                -
//          ,ELE = COBECHO@
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                -
//          ,ELE = AREA1
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                -
//          ,ELE = AREA2
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                -
//          ,ELE = AREA3
//MODIFY-SYMBOL-ATTR   PUBLIC = YES
//END-SUB-LLM
//REMARK ------------------------------------------------------------
//REMARK ------------------------------------------------------------
//SET-EXTERN-RESOLUTION  SYMBOL-TYPE=REFERENCE , RESOLUTION=STD
//MODIFY-MAP-DEFAULT  PROGRAM-MAP = PAR( DEFINITION = ALL           -
//                                  ,INVERT         = ALL           -
//                                  ,REFERENC       = ALL)          -
//                                ,UNRESOLVED     = YES             -
//                                ,SORTED-PRO     = YES             -
//                                ,DUPLICATE      = YES             -
//                                ,OUTPUT= LIST.LINK.EXAMPLE-LLM
/REMARK ------------------------------------------------------------
//SAVE-LLM   LIB = LIB.LOAD-MODULE.STARTUP                          -
//                ,ELEM                = *INTERNAL                  -
//                ,FOR-BS2000-VERSIONS = *FROM-OSD-V4(...)
//REMARK ------------------------------------------------------------
//END
```

Wie das Beispiel zeigt, reicht es, alle shareable Module, die nicht das PUBLIC-Attribut besitzen, in dem Binder-Lauf in ein SUB-LLM zu binden, und dann für das ganze SUB-LLM das Attribut PUBLIC mit der Anweisung MODIFY-SYMBOL-ATTRIBUTES zu setzen. In der Liste LIST.LINK.EXAMPLE-LLM - Abschnitt *PROGRAM MAP* - kann man sich dann anschauen, in welchem Slice sich die einzelnen Objekte befinden.

3.2.4 Laufzeitsysteme binden

Viele Laufzeitsysteme besitzen Teile, die shareable sind. Dies kann zusammen mit BLS dazu benutzt werden, Ladezeiten zu verkürzen und Performance zu gewinnen. Dabei gibt es je nach Laufzeitsystem die folgenden Möglichkeiten:

- Shareable Teile als Subsystem laden. Dies ist die beste Möglichkeit, die aber nicht für alle Laufzeitsysteme möglich ist.
- Shareable Teile vorbinden und in einen Common Memory Pool laden.
- Die benötigten Teile des Laufzeitsystems zu einem LLM binden und dann dynamisch laden.
- Das Laufzeitsystem statisch zum Start-LLM binden.

Bitte beachten Sie beim Binden einer UTM-Anwendung mit SYSLNK.CRTE.PARTIAL-BIND, dass die CRTE-Version auf dem System, auf dem die Anwendung gebunden wird, nicht höher sein darf als die CRTE-Version auf dem/n System/en, auf dem/denen die Anwendung zum Ablauf kommt.

Beim Binden des Start-LLMs müssen für openUTM das C- und SPL-Laufzeitsystem sowie das ILCS eingebunden werden (ILCS ist in den CRTE-Bibliotheken vorhanden). Dies erreichen Sie, indem Sie die beiden folgenden Binderanweisungen in dieser Reihenfolge geben:

```
//RESOLVE-BY-AUTOLINK LIB=$userid.SYSLNK.CRTE[.PARTIAL-BIND]
```

```
//RESOLVE-BY-AUTOLINK LIB=$userid.SYSLNK.UTM.070.SPLRTS
```

Wenn Sie Ihre Laufzeitsysteme statisch binden wollen, reichen diese beiden Anweisungen für C-, COBOL- und SPL-Objekte aus.

Wenn Sie weitere Laufzeitsysteme benötigen, dann müssen Sie - sofern nichts anderes im zugehörigen Handbuch steht - die RESOLVE-BY-AUTOLINK-Anweisungen für die benötigten Laufzeitsysteme zwischen die RESOLVE-Anweisungen für CRTE und UTM einfügen.

Besser ist es, zum Binden der Laufzeitsysteme die Partial-Bind-Bindetechnik zu verwenden. Das Binden mit der Bibliothek SYSLNK.CRTE.PARTIAL-BIND hat den Vorteil, dass Binde- und Ladezeit verkürzt werden und dass das gebundene Programm weniger Plattenspeicher belegt. Beim Binden mit der Partial-Bind-Bibliothek werden Referenzen in das Laufzeitsystem durch Verbindungsmodule befriedigt. Die beim Ablauf benötigten Module des Laufzeitsystems werden erst zum Ablaufzeitpunkt dynamisch nachgeladen bzw. aus dem Sharecode verwendet.

Falls Sie SYSLNK.CRTE oder SYSLNK.CRTE.PARTIAL-BIND - neben dem Binden des Start-LLM - ein weiteres Mal benötigen, dann müssen Sie die jeweilige CRTE-Bibliothek nochmals verwenden. Ein solcher Fall ist das Binden des tasklokalen Teils des COBOL-Laufzeitsystems, siehe Beispiel im Abschnitt "Laufzeitsysteme zu einem LLM binden".

Die Möglichkeiten für das shareable Laden werden in den Folgeabschnitten erläutert. Informationen darüber, welche Laufzeitroutinen shareable und welche nicht-shareable sind, können Sie der Beschreibung des jeweiligen Laufzeitsystems entnehmen.

Beachten Sie bitte, dass Sie ein Lademodul, das Module der Laufzeitsysteme enthält, nie während des laufenden Anwendungsprogramms austauschen dürfen, da dies zu schlecht diagnostizierbaren Fehlern führen kann.

3.2.4.1 Shareable LZS-Teile als Subsystem

Wenn Ihre UTM-Anwendung mit der gemeinsamen Laufzeitumgebung CRTE (Common Run Time Environment) abläuft, so besteht die Möglichkeit, die shareable Teile von CRTE über DSSM als Subsystem zu laden. Auch andere Laufzeitsysteme bieten diese Möglichkeit an.

Das Laden des Subsystems ist Aufgabe des BS2000-Systemverwalters, näheres hierzu ist im Handbuch CRTE beschrieben.

Wenn möglich, sollten Sie von diesem Verfahren Gebrauch machen, da es Systemressourcen schont. Die Laufzeitsysteme für C und COBOL, die in CRTE enthalten sind, sollten jedoch unbedingt als CRTE Subsystem geladen werden, da sie auch von KDCROOT benötigt werden.

Wenn das CRTE-Laufzeitsystem vorgeladen ist, dann können Sie dieses nutzen, indem Sie beim Binden die Bibliothek SYSLNK.CRTE.PARTIAL-BIND verwenden. Die Verbindung zum CRTE-Subsystem wird dann durch ein Verbindungsmodul hergestellt, das beim Binden des Start-LLM eingebunden wird. Ist das CRTE-Laufzeitsystem nicht erreichbar, dann lädt das Verbindungsmodul die notwendigen Teile aus der CRTE-Bibliothek in den tasklokalen Teil nach.

3.2.4.2 Shareable LZS-Teile in Common Memory Pools

Wenn Module ein Laufzeitsystem benutzen, das nicht als Subsystem geladen werden kann und das auch nicht im Start-LLM eingebunden ist, dann sollten Sie alle shareable Module des Laufzeitsystems jeweils einer Programmiersprache zu LLMs vorbinden und durch openUTM in einen Common Memory Pool laden.

Die shareable Teile des Laufzeitsystems müssen Sie dann wie folgt generieren:

```
LOAD-MODULE share-rts-part,LOAD-MODE=(POOL,..., NO-PRIVATE-SLICE),...
```

Die nicht-shareable Teile des Laufzeitsystems (ebenfalls zu einem LLM vorgebunden) müssen Sie ebenfalls generieren:

```
LOAD-MODULE nonshare-rts-part,LOAD-MODE=STARTUP
```

Die UTM-Generierung wird einfacher, wenn Sie die shareable und die nicht-shareable Teile zu einem LLM zu binden. Dieses Lademodul müssen Sie generieren mit:

```
LOAD-MODULE rts,LOAD-MODE=(POOL,..., STARTUP),...
```

Beim Generieren müssen Sie darauf achten, dass dieses LLM **vor** den anderen Modulen angegeben wird, die das Laufzeitsystem benötigt, da es sonst zu unbefriedigten Externverweisen führt.
Beachten Sie bitte, dass shareable Teile des C-, des SPL- und des FOR1-Laufzeitsystems nicht in einen von openUTM verwalteten Common Memory Pool geladen werden dürfen.

3.2.4.3 Laufzeitsysteme zu einem LLM binden

Für alle Laufzeitsysteme, die keine PARTIAL-BIND-Bibliothek besitzen, müssen Sie feststellen, welche Module des Laufzeitsystems für die gesamte Anwendung benötigt werden. Dazu gehen Sie folgendermaßen vor:

1. Alle Anwendungsteilprogramme und alle Event-Exits der gewünschten Sprache sowie alle shareable Module des Laufzeitsystems, die als Subsystem oder in einen Common Memory Pool geladen sind, müssen zu einem Sub-LLM zusammengebunden werden.
2. Das ILCS (enthalten in CRTE) muss ebenfalls in das Sub-LLM eingebunden werden, und zwar die höchste zur Verfügung stehende Version.
3. Um alle Externverweise auf das Laufzeitsystem zu befriedigen, muss die RESOLVE-BY-AUTOLINK-Anweisung auf die Bibliothek des Laufzeitsystems der Sprache eingegeben werden: RESOLVE-BY-AUTOLINK LIBRARY=*rts-bibliothek*.
4. Mit Hilfe der Anweisung REMOVE-MODULES müssen Sie dann das Sub-LLM (siehe 1. und 2.) wieder aus dem LLM entfernen.
5. Das Lademodul generieren Sie in openUTM mit folgender Anweisung:

```
LOAD-MODULE RTS-PRIVATE-KERNEL   -
            ,VERSION=001         -
            ,LIB=private-rts-lib -
            ,LOAD-MODE=STARTUP
```

3.2.5 Start-LLM binden

Beim Binden des Start-LLM wird zwischen den folgenden Möglichkeiten unterschieden:

- Sie binden das ROOT-Tabellenmodul statisch in das Start-LLM mit ein oder
- das ROOT-Tabellenmodul wird dynamisch beim Start des Anwendungsprogramms nachgeladen. Dazu müssen Sie in den Startparametern für TABLIB den Namen der Bibliothek, die das ROOT-Tabellenmodul enthält, und für ROOTNAME den Namen des ROOT-Tabellenmoduls (PLAM-Elementname) angeben (siehe Abschnitt „Startparameter für openUTM“). Den Modul KDCRTMN müssen Sie als ersten Modul statisch einbinden.

Das zweite Verfahren ist vorteilhafter, da Sie bei Änderungen in der UTM-Generierung die Anwendung nicht neu binden müssen. Aus dem gleichen Grund sollten Sie möglichst alle Teilprogramme in Lademodule ablegen, die dynamisch gebunden werden.

Bei beiden Verfahren müssen Sie die folgenden Punkte beachten:

- Die Laufzeitsysteme für C und SPL zusammen mit dem ILCS werden von den Modulen des KDCROOT benötigt, weswegen Sie die nicht-shareable Teile des Laufzeitsystems immer statisch einbinden müssen.
- Das Start-LLM darf nach dem Binden nur unbefriedigte Externverweise auf solche Module der Laufzeitsysteme haben, die entweder vor dem Anwendungsstart als Subsystem geladen sind oder die beim Anwendungsstart nachgeladen werden.
- Sie sollten vermeiden, mehrere Start-LLM in der gleichen Bibliothek abzulegen, da BLS beim Starten versucht, offene Externverweise aus dieser Bibliothek zu befriedigen. D.h. neben Lademodulen, die dynamisch nachgeladen werden, sollte die Bibliothek nur ein Start-LLM enthalten, das mit dem Kommando START-EXECUTABLE-PROGRAM geladen wird. Die AUTOLINK-Funktion können Sie über Parameter von START-EXECUTABLE-PROGRAM auch auf bestimmte Bibliotheken eingrenzen oder komplett ausschalten.
- Die RESOLVE-Anweisung auf die Bibliothek SYSLNK.CRTE.PARTIAL-BIND bzw. SYSLNK.CRTE müssen Sie stets als erste RESOLVE-Anweisung auf eine Laufzeitsystem-Bibliothek und nach der RESOLVE-Anweisung auf die UTM-Bibliothek angeben. Die Bibliothek, die Sie im Start-LLM zum Binden der benötigten Module des CRTE verwenden, müssen Sie auch beim Binden aller Lademodule der Anwendung angeben. Die Bibliothek SYSLNK.CRTE.PARTIAL-BIND bringt Performancevorteile und sollte daher bevorzugt eingesetzt werden.

Beispiel

Der folgende BINDER-Lauf enthält alle Anweisungen, die zum Binden eines Start-LLM benötigt werden. Es wird davon ausgegangen, dass die Anwendung mit der gemeinsamen Laufzeitumgebung CRTE abläuft und UTM und CRTE über IMON installiert wurden.

Der String *vvv* steht dabei für die openUTM-Version (z.B. 070 für V7.0).

```
/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=start-llm                          -
//                  ,INTERNAL-VERSION=start-llm-version
//REMARK  +-----------------------------------------------------------+
//BEGIN-SUB-LLM-STATEMENTS SUB-LLM-NAME=ROOT-TAB-LLM ---------------------- 1
//INCLUDE-MODULES LIBRARY=tablib ,ELEMENT=root-module
//END-SUB-LLM-STATEMENTS
//REMARK  +-----------------------------------------------------------+
//INC-MOD ELEM=KDCRTMN ,LIB=<userid1>.SYSLNK.UTM.vvv ---------------------- 2
//REMARK  +-----------------------------------------------------------+
//BEGIN-SUB-LLM-STATEMENTS SUB-LLM-NAME=LM-SHARED-RTS --------------------- 3
//INCLUDE-MODULES LIBRARY=$userid2.SYSLNK.CRTE.PARTIAL-BIND, ELEMENT=ITCMADPT
//INCLUDE-MODULES oncall-load-module
//INCLUDE-MODULES startup-load-module
//INCLUDE-MODULES pool-load-module
//END-SUB-LLM-STATEMENTS
//REMARK  +-----------------------------------------------------------+
//RESOLVE-BY-AUTOLINK LIBRARY=$userid1.SYSLNK.UTM.vvv --------------------- 4
//RESOLVE-BY-AUTOLINK LIBRARY=user-lib
//REMARK  +-----------------------------------------------------------+
//RESOLVE-BY-AUTOLINK LIBRARY=$userid2.SYSLNK.CRTE.PARTIAL-BIND ----------- 5
//RESOLVE-BY-AUTOLINK LIBRARY=other-rts-lib
//RESOLVE-BY-AUTOLINK LIBRARY=$userid1.SYSLNK.UTM.vvv.SPLRTS
//REMARK  +-----------------------------------------------------------+
//SHOW-MAP ..., UNRESOLVED-LIST=SORTED, ... ------------------------------- 6
//REMARK  +-----------------------------------------------------------+
//REMOVE-MODULES NAME=*ALL,PATH-NAME=.ROOT-TAB-LLM ----------------------- 7
//REMOVE-MODULES NAME=*ALL,PATH-NAME=.LM-SHARED-RTS ----------------------- 8
//REMARK  +-----------------------------------------------------------+
//SET-EXTERN-RESOLUTION SYMBOL-TYP=REFERENCES,RESOLUTION=STD
//SAVE-LLM LIBRARY=start-bibliothek,ELEMENT=*INTERNAL-NAME(           -
//                                  VERSION=*INTERNAL-VERSION)        -
//       ....
//END
```

1. Als erstes wird immer das ROOT-Tabellenmodul in ein eigenes Sub-LLM gebunden. Dieses Modul können Sie vor dem Abspeichern des Start-LLM wieder entfernen (siehe 7), falls das ROOT-Tabellenmodul, wie hier im Beispiel, nicht statisch eingebunden, sondern beim Start des Anwendungsprogramms dynamisch nachgeladen werden soll.

   Durch die Externverweise des ROOT-Tabellenmoduls werden die statischen Teilprogramme aus *user-lib* dazugebunden (siehe 4). Dies sollten allerdings möglichst wenige Teilprogramme sein, da diese Module nicht ausgetauscht werden können.

2. Als nächstes wird der ROOT-Systemmodul KDCRTMN gebunden. Dieser muss nur dann explizit gebunden werden, falls ein Start-LLM ohne ROOT-Tabellenmodul erzeugt werden soll.

3. Dann wird in einem eigenen Sub-LLM ein Adapter-Modul (ITCMADPT) gebunden, der dafür sorgt, dass die benötigten Laufzeit-Module nachgeladen werden.

   Zusätzlich werden alle vorhandenen Lademodule, die dynamisch nachgeladen werden, in dieses Sub-LLM gebunden, damit die benötigten Module des Laufzeitsystems durch die RESOLVE-BY-AUTOLINK-Anweisung zum Start-LLM hinzugebunden werden. Dadurch sind sie beim dynamischen Nachladen schon vorhanden und brauchen nicht mehr über die Autolink-Funktion nachgeladen werden, weil dies die Ladeperformance verschlechtert.

Wenn Sie das COBOL-Subsystem benutzen, kann der Adapter-Modul (ITCMADPT) aus dem oben genannten SUB-LLM entfallen.

4. Als erste Bibliothek in einer RESOLVE-BY-AUTOLINK-Anweisung müssen Sie stets die UTM-Bibliothek angeben. Danach können RESOLVE-BY-AUTOLINK-Anweisungen auf Benutzerbibliotheken folgen.
5. Als erste Bibliothek eines Laufzeitsystems in einer RESOLVE-BY-AUTOLINK-Anweisung müssen Sie stets die CRTE-Bibliothek angeben. Nach der CRTE-Bibliothek sollten Sie die RESOLVEs auf eventuell vorhandene Bibliotheken anderer Laufzeitsysteme eingeben, und zwar vor der Bibliothek mit dem von openUTM benötigten SPL-Laufzeitsystem. Weitergehende Informationen sind in dem Handbuch des entsprechenden Laufzeitsystems zu finden.
6. Mit der SHOW-MAP-Anweisung können Sie sich u.a. die nicht aufgelösten Externverweise auflisten lassen. Die Liste, die an dieser Stelle des BINDER-Laufs erstellt wird, enthält alle offenen Externverweise, die noch befriedigt werden müssen. Am Ende des BINDER-Laufs erhalten Sie automatisch eine weitere Liste der unbefriedigten Externverweise, die aber zusätzlich alle die unbefriedigten Externverweise enthält, die durch das Entfernen des Sub-LLM entstehen.
7. Mit einer REMOVE-MODULES-Anweisung entfernen Sie das ROOT-Tabellenmodul aus dem Start-LLM.
8. Auch der (falls nötig) eingebundene Adaptermodul (ITCMADPT) sowie die shareable Module des Laufzeitsystems für COBOL und die Lademodule, die mit ONCALL, STARTUP oder POOL generiert sind, werden aus dem Start-LLM entfernt.

Das so gebundene Start-LLM muss mit dem folgenden Kommando geladen werden:

```
/START-EXECUTABLE-PROGRAM FROM-FILE=*LIBRARY-ELEMENT     -
/                     (LIBRARY=start-library              -
/                     ,ELEMENT-OR-SYMBOL=start-llm)       -
/           ,DBL-PARAMETERS=*PARAMETERS(                  -
/           ,LOADING=*PARAMETERS(                         -
/                      PROGRAM-MODE = *ANY                -
/                      ,LOAD-INFORMATION = *REFERENCES))  -
/           ,ERROR-PROCESSING=*PARAMETERS(                -
/                      UNRESOLVED-EXTRNS=*DELAY           -
/                      ,ERROR-EXIT = *NONE)
```

3.3 Hinweise für Anwendungen mit ILCS-Teilprogrammen

- Programmübergänge (CALL Aufrufe) zwischen ’ILCS’-Teil- oder -Unterprogrammen und ’Nicht-ILCS’-Unterprogrammen sind verboten.
- ILCS-Laufzeitsystem-Module (Präfix IT0....) dürfen im geladenen Anwendungsprogramm nur einmal enthalten sein. D.h. Sie müssen verhindern, dass die ILCS-Module beim Binden per RESOLVE aus den RTS-Bibliotheken der Compiler mehrfach hinzugebunden werden oder dass sie durch dynamisches Nachladen von Modulen mehrfach geladen werden.
- Es muss immer die höchste zur Verfügung stehende ILCS-Version geladen werden. In der CRTE-Bibliothek ist immer ein ILCS enthalten.
- Ob die betreffende Compilerversion bzw. das Laufzeitsystem ILCS unterstützt und welche ILCS-Version, ist dem Benutzerhandbuch bzw. Serviceinformationen wie Freigabemitteilung und Readme-Dateien des Compilers zu entnehmen.

  Sie finden im Abschnitt "Compiler-Versionen, Laufzeitsysteme, KDCDEF-Optionen" eine detaillierte Aufstellung mit den möglichen Kombinationen von Compiler-Option und RTS.
- Beispiele zum Binden von ILCS-Programmen sind im CRTE-Handbuch zu finden.

4 Für den Betrieb notwendige Dateien

Vor jedem Starten einer UTM-Anwendung müssen Sie dafür sorgen, dass die folgenden, für den Betrieb einer UTM-Anwendung notwendigen Dateien vorhanden sind:

- die KDCFILE
- die Systemdateien SYSOUT und SYSLST sind immer vorhanden, sollten aber Prozess-spezifischen realen Dateien zugewiesen werden
- die System-Protokolldatei SYSLOG
- die Benutzer-Protokolldatei(en) USLOG (optional)
- alle Programm- und Objektmodul-Bibliotheken, aus denen beim Starten und während des Betriebs der Anwendung Module dynamisch nachgeladen werden sollen.

KDCFILE, USLOG und SYSLOG (falls ohne Linkname) haben das Präfix *filebase* (=Basisname der UTM-Anwendung). *filebase* müssen Sie in den Startparametern angeben. Den Namen *filebase* legen Sie beim Erzeugen der KDCFILE mit dem Generierungstool KDCDEF fest, siehe openUTM-Handbuch „Anwendungen generieren“, Steueranweisung (MAX...,KDCFILE=*filebase*).

4.1 Systemdateien SYSOUT und SYSLST

openUTM protokolliert Meldungen auf die Systemdateien SYSOUT und/oder SYSLST, d.h. openUTM schreibt alle Meldungen mit Meldungsziel SYSOUT oder SYSLST in diese Dateien (zu „Meldungsziel“ siehe openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“). Diese Systemdateien sollten daher auf eigene Prozess-spezifische Dateien gelegt werden.

Diese Systemdateien können Sie im laufenden Betrieb manuell oder automatisch auf neue Dateien umschalten. Nach dem Umschalten können die alten SYSOUT- und SYSLST-Dateien ausgewertet und ggf. gelöscht werden, um den belegten Plattenspeicherplatz zu reduzieren.

Nach den BS2000-Kommandos SET-SYSOUT-READ-MARK bzw. SET-SYSLST-READ-MARK können Sie auch auf die noch geöffneten Systemdateien zugreifen (siehe Handbuch „BS2000 OSD/BC Kommandos“).

Systemdateien umschalten

Die Systemdateien können im laufenden Betrieb per Administration oder über ein einstellbares Zeitintervall umgeschaltet werden. Die Systemdateien aller Tasks einer UTM-Anwendung werden immer alle gemeinsam umgeschaltet, der Umschaltzeitpunkt kann sich aber unter Last für einzelne Tasks verzögern.

- Per Administration schalten Sie die Systemdateien um
  - mit dem Kommando KDCAPPL SYSPROT=NEW
  - per Programmschnittstelle mit dem Feld *sysprot_switch* in der Datenstruktur *kc_diag_and_account_par_str* (siehe openUTM-Handbuch „Anwendungen administrieren“)
  - über WinAdmin/WebAdmin

  Die Systemdateien werden möglichst zeitnah zur Aufforderung umgeschaltet.
- Um die Systemdateien zeitgesteuert umzuschalten, geben Sie beim Starten der UTM-Anwendung den Startparameter SYSPROT an (siehe Abschnitt „Startparameter für openUTM“). Sie können hier ein Zeitintervall in Tagen angeben. Das Umschalten erfolgt jeweils um Mitternacht.

  Tritt beim Umschalten ein Fehler auf, wird eine Fehlermeldung ausgegeben und das zeitgesteuerte Umschalten deaktiviert.

Namen der umgeschalteten Dateien

Beim Starten der UTM-Anwendung werden die Systemdateien mit den vom System oder vom Anwender festgelegten Namen eingerichtet. Beim ersten manuellen oder automatischen Umschalten werden die Dateinamen gemäß folgender Formate erzeugt:

SYSOUT: < `prefix>.O.YYMMDD.HHMMSS.TSN`
SYSLST: < `prefix>.L.YYMMDD.HHMMSS.TSN`

Die Gesamtlänge des neuen Pfadnamens für eine umgeschaltete Datei beträgt maximal 54 Zeichen und setzt sich wie folgt zusammen:

```
:catid:$userid.filename-prefix.datei-suffix

<---6---><---10---><------17-----><----21---->
```

<prefix>

Das Präfix setzt sich zusammen aus

catid, userid

Katalog- und Benutzerkennung, unter der die UTM-Anwendung gestartet wurde.

filename-prefix

Dateinamens-Präfix, das Sie beim Starten der UTM-Anwendung für den Startparameter SYSPROT angegeben haben (siehe Abschnitt „Startparameter für openUTM“).

Das Dateinamens-Präfix darf höchstens 17 Zeichen lang sein.

Standardwert für <filename-prefix> :
Name der Anwendung, der bei der KDCDEF-Generierung in MAX APPLINAME festgelegt wurde.

YYMMDD.HHMMSS

Datum und Uhrzeit des Umschaltzeitpunkts

TSN TSN der Task

4.2 System-Protokolldatei SYSLOG

openUTM protokolliert Ereignisse aus dem Lauf der Anwendung in die System-Protokolldatei SYSLOG (**SYS**TEM **LOG**GING), d.h. openUTM schreibt alle Meldungen mit Meldungsziel SYSLOG in diese Datei (zu „Meldungsziel“ siehe openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“).

Die System-Protokolldatei SYSLOG können Sie für die laufende Überwachung des Anwendungslaufs oder für spätere Kontrollen nutzen. Insbesondere für die Diagnose liefert die SYSLOG-Datei wichtige Information.

Sie müssen jeder UTM-Anwendung eine SYSLOG-Datei zur Verfügung stellen. Wenn Sie einer UTM-Anwendung vor dem Start keine SYSLOG-Datei zugewiesen haben, dann wird der Start der Anwendung abgebrochen.

Die System-Protokolldatei SYSLOG können Sie anlegen als:

- einfache Datei SYSLOG
- Dateigenerationsgruppe SYSLOG-FGG (**F**ile **G**eneration **G**roup)

Gegenüber einer einfachen SYSLOG-Datei hat eine SYSLOG-FGG folgende Vorteile:

- Sie können im laufenden Betrieb der Anwendung auf die jeweils folgende Dateigeneration umschalten (umschaltbare SYSLOG-Datei). Wenn alle Tasks der Anwendung die alte SYSLOG-Datei geschlossen haben, können Sie frei über diese Dateigeneration verfügen.Die SYSLOG können Sie z.B. mit dem Administrationskommando KDCSLOG administrieren. Sehen Sie dazu openUTM-Handbuch „Anwendungen administrieren“.
- Sie können eine automatische Größenüberwachung für die SYSLOG einstellen. D.h. Sie können einen Schwellwert für die Größe der einzelnen Dateigenerationen der SYSLOG-FGG generieren bzw. per Administration festlegen, bei dem openUTM automatisch auf die folgende Dateigeneration der FGG umschaltet. Die Größenüberwachung kann während des Anwendungsbetriebs ein- und ausgeschaltet werden.

Meldungen von openUTM

openUTM gibt die folgenden Meldungen bezüglich der SYSLOG aus:

- Beim Anwendungsstart die Meldung K136:

  ```
  K136 (Erste) SYSLOG-Datei ist <dateiname>
  ```

- Beim Anwendungsende die Meldung K138:

  ```
  K138 SYSLOG-Datei <dateiname> geschlossen
  ```

- Beim Umschalten auf eine andere Dateigeneration die Meldung K137:

  ```
  K137 SYSLOG umgeschaltet auf Datei <dateiname>
  ```

4.2.1 SYSLOG als einfache Datei

Wenn die SYSLOG-Datei als einfache Datei geführt werden soll, dann können Sie diese openUTM auf zwei Arten bekannt machen:

- Vor jedem Start der Anwendung weisen Sie der Datei den Linknamen SYSLOG zu (SET-FILE-LINK-Kommando). Beim Start der Anwendung öffnet openUTM diese Datei. Sie bleibt während des gesamten Anwendungslaufs für alle Tasks der Anwendung geöffnet.
- Sie können auch vor dem Start eine Datei mit dem Namen *filebase*.SLOG anlegen. Die Datei muss dieselbe Katalogkennung (CATID) haben wie die KDCFILE *filebase*.KDCA (siehe openUTM-Handbuch „Anwendungen generieren“, Anweisung MAX...,CATID und KDCFILE). openUTM benutzt die Datei mit dem Namen *filebase*. SLOG bei jedem folgenden Start als SYSLOG-Datei, solange beim Start der Anwendung keine Datei bzw. Dateigeneration mit dem Linknamen SYSLOG existiert. Ist eine Datei bzw. Dateigeneration mit dem Linknamen SYSLOG vorhanden, dann protokolliert openUTM die SYSLOG-Meldungen immer in dieser Datei.

Nach dem Ende eines Anwendungslaufs sollten Sie den Inhalt der aktuellen SYSLOG-Datei sichern. Beim folgenden Start der Anwendung wird der Inhalt dieser Datei von openUTM überschrieben.

> **ACHTUNG!**
>
> Wollen Sie die SYSLOG als einfache Datei führen, dann dürfen Sie die Größenüberwachung für die SYSLOG **nicht** generieren. Wenn Sie in diesem Fall die Größenüberwachung bei der UTM-Generierung einschalten mit MAX...,SYSLOG-SIZE=*size* (*size* > 0), bricht openUTM den Start der Anwendung mit Startfehlercode 58 ab.

4.2.2 Dateigenerationsgruppe SYSLOG-FGG

Wenn die SYSLOG-Datei als FGG geführt wird, dann öffnet openUTM beim Start der Anwendung eine Dateigeneration der SYSLOG-FGG als SYSLOG-Datei. Alle Tasks der Anwendung schreiben die Meldungen mit Ziel SYSLOG zunächst in diese Dateigeneration.

Wird die SYSLOG als Dateigenerationsgruppe angelegt, dann reicht es, wenn Sie vor dem Start die Dateigenerationsgruppe (FGG) anlegen. Die einzelnen Dateigenerationen müssen Sie nicht anlegen, weil openUTM vor dem Öffnen einer SYSLOG-Dateigeneration prüft, ob diese schon existiert. Existiert die Dateigeneration, dann wird sie lediglich geöffnet. Existiert die Dateigeneration noch nicht, dann legt openUTM die Dateigeneration selbstständig an, mit den Merkmalen der Dateigenerationsgruppe.

Wollen Sie den Dateigenerationen bestimmte Werte für Primary und Secondary Allocation zuordnen, dann müssen Sie eine Dateigeneration der FGG mit den gewünschten Werten anlegen. openUTM legt dann die nächsten Dateigenerationen mit denselben Werten an.

SYSLOG-FGG bekanntmachen

Sie haben die folgenden Möglichkeiten openUTM die SYSLOG-FGG bekannt zu machen:

- Verwendung des Linknamens „SYSLOG“

  Sie legen vor dem Start der Anwendung eine FGG mit beliebigem Namen für die SYSLOG an. Ordnen Sie dann einer Dateigeneration der FGG mit /SET-FILE-LINK den Linknamen SYSLOG zu. Die Zuweisung des Linknamens muss vor jedem Start der Anwendung erfolgen.

  In diesem Fall wird die Dateigenerationsgruppe, zu der die Dateigeneration mit dem Linknamen SYSLOG gehört, als SYSLOG-FGG genommen. openUTM protokolliert zunächst in die Dateigeneration mit dem Linknamen SYSLOG (erste SYSLOG-Dateigeneration).

  Haben Sie für Ihre Anwendung die automatische Größenüberwachung generiert, dann schaltet openUTM auf die nächste Dateigeneration dieser FGG um, sobald die Größe der ersten SYSLOG-Dateigeneration den Schwellwert der Größenüberwachung erreicht hat. Existiert diese Dateigeneration noch nicht, legt openUTM sie selbst an.

  Wurde die Anwendung ohne automatische Größenüberwachung generiert, dann protokolliert openUTM solange in die Dateigeneration mit dem Linknamen SYSLOG, bis Sie auf eine andere Dateigeneration der FGG weiterschalten oder die automatische Größenüberwachung einschalten (z.B. mit dem Administrationskommando KDCSLOG). Diese Dateigeneration wird ebenfalls beim Umschalten von openUTM automatisch angelegt.

- Vor dem ersten Start der Anwendung eine FGG mit dem Namen *filebase*.SLOG anlegen
  Diese Dateigenerationsgruppe muss den gleichen Basisnamen (einschließlich CATID und USERID) haben wie die KDCFILE (KDCA-Datei) und sie muss unter der gleichen BS2000-Benutzerkennung eingerichtet sein, unter der die UTM-Prozesse laufen.
  openUTM benutzt die FGG mit dem Namen *filebase*.SLOG nur als SYSLOG-FGG, wenn beim Start der Anwendung keine Datei bzw. Dateigeneration unter der Benutzerkennung der Anwendung vorhanden ist, der der Linkname SYSLOG zugeordnet ist. Ist eine Datei bzw. Dateigeneration mit dem Linknamen SYSLOG vorhanden, dann protokolliert openUTM die SYSLOG-Meldungen immer in dieser Datei.

Ist *filebase*.SLOG eine Dateigenerationsgruppe, dann entscheidet die eingestellte Basis der FGG, welche Dateigeneration als erste SYSLOG-Datei genommen wird.

Basis außerhalb des gültigen Bereichs
Liegt die Basis außerhalb des gültigen Bereichs (z.B. BASE-NUM=0), dann legt openUTM beim Start der Anwendung die Dateigeneration mit der Generationsnummer LAST-GEN+1 an. Diese Dateigeneration ist dann die erste SYSLOG-Datei.
Beim nächsten Start der Anwendung nimmt openUTM als erste SYSLOG-Datei die Folgegeneration der im vorangegangenen Anwendungslauf zuletzt beschriebenen Dateigeneration. D.h. wurde im letzten Anwendungslauf bis zur *n*-ten Dateigeneration geschrieben, dann wird beim nächsten Start in der (*n*+1)-ten Dateigeneration begonnen.

Basis innerhalb des gültigen Bereichs
Liegt die Basis innerhalb des gültigen Bereichs zwischen der ersten und der letzten Dateigeneration (in der Ausgabe des Kommandos SHOW-FILE-ATTRIBUTES auf die FGG ist: FIRST-GEN <= BASE-NUM <= LAST-GEN), dann wird die Basisgeneration als erste SYSLOG-Datei genommen.

openUTM verändert die von Ihnen eingestellte Basisnummer nicht. Beim nächsten Start der Anwendung beginnt openUTM mit der Protokollierung wieder in derselben Dateigeneration wie beim vorherigen Start, es sei denn, Sie haben zuvor die Basiseinstellung geändert.

4.2.2.1 SYSLOG-FGG anlegen

Wollen Sie mit einer SYSLOG-FGG arbeiten, dann müssen Sie diese vor dem Start der Anwendung anlegen.

SYSLOG als *filebase*.SLOG anlegen

Im einfachsten Fall arbeiten Sie mit der FGG *filebase*.SLOG, bei der die Basis außerhalb des gültigen Bereichs liegt. Diese FGG müssen Sie nur einmal vor dem ersten Start der Anwendung anlegen. Bei jedem weiteren Anwendungsstart macht openUTM automatisch mit der Folgegeneration zur zuletzt beschriebenen Dateigeneration weiter, sofern Sie die Basis nicht in den gültigen Bereich verlegen. Sie können die FGG mit dem folgenden BS2000-Kommando anlegen:

```
/CREATE-FILE-GROUP                                  -
/   GROUP-NAME = filebase.SLOG                      -
/  ,GENERATION-PARAMETER = *GENERATION-PARAMETER(   -
/      MAXIMUM = n                                  -
/    [,VOLUME = volume                              -
/     ,DEVICE-TYPE = device                         -
/     ,OVERFLOW-OPTION = overflow ] )
```

Bedeutung der Parameter:

MAXIMUM=n

maximale Anzahl der Dateigenerationen, die in der FGG gleichzeitig katalogisiert sein dürfen.

VOLUME=volume, DEVICE=device

Datenträgerkennzeichen und Gerätetyp der Platte, auf der die FGG eingerichtet werden soll. Die FGG kann auf PUBLIC oder PRIVATE DISK eingerichtet werden.

OVERFLOW-OPTION=overflow

gibt an, was geschehen soll, wenn die maximal erlaubte Anzahl an Dateigenerationen (MAXIMUM) überschritten wird. Über diesen Operanden können Sie steuern, ob von der SYSLOG-FGG Ihrer Anwendung nur die letzten *n* (MAXIMUM) Dateigenerationen erhalten bleiben sollen oder alle von openUTM beschriebenen Dateigenerationen. Sie können CYCLIC-REPLACE, REUSE-VOLUME oder KEEP-GENERATION angeben. Beachten Sie dazu auch Abschnitt „Kennungs-Überlaufschutz“ und Abschnitt „SYSLOG-Generationen erhalten“.

KEEP-GENERATION

alle Dateigenerationen bleiben erhalten, auch wenn die Anzahl in MAXIMUM überschritten wird.

CYCLIC-REPLACE und REUSE-VOLUME

bewirken, dass die älteste Dateigeneration der FGG vor dem Anlegen der neuen gelöscht wird.

Geben Sie hier CYCLIC-REPLACE oder REUSE-VOLUME an, dann sollten Sie den Wert für MAXIMUM (Anzahl der Dateigenerationen) nicht zu klein wählen. Nach dem Umschalten kann die „alte“ Dateigeneration, die von openUTM nicht mehr beschrieben wird, noch über längere Zeit von einigen Tasks offengehalten werden (wenn diese sich sehr lange mit einem Anwender-Teilprogramm beschäftigen). Haben Sie nun *n* Dateigenerationen zugelassen und hält eine Task noch die Dateigeneration *i* offen, dann kann noch nicht auf die Dateigeneration *i+n* umgeschaltet werden. Das BS2000-System meldet einen DMS-Fehler für diese Dateigeneration. Die automatische Größenüberwachung wird suspendiert, bis der Administrator der Anwendung mit dem Kommando KDCSLOG die SYSLOG-Datei erfolgreich umschaltet.

Bei OVERFLOW-OPTION=KEEP-GENERATION ist jedoch auch in diesem Fall ein Umschalten möglich.

Mit dem Kommando legen Sie eine FGG an, deren Basis auf 0 liegt, d.h. außerhalb des gültigen Bereichs. openUTM legt alle Dateigenerationen selbst an. Beim ersten Start der Anwendung legt openUTM die Dateigeneration mit der Generationsnummer 1 an und benutzt diese als erste SYSLOG-Datei.

SYSLOG mit Linknamen anlegen

Wollen Sie mit dem Linknamen SYSLOG arbeiten, dann können Sie die SYSLOG mit demselben Kommando wie bei einer SYSLOG mit *filebase*.SLOG erzeugen, siehe oben. Für GROUP-NAME können Sie einen beliebigen Namen *fgg-name* angeben. Vor jedem Start der Anwendung müssen Sie dann der Dateigeneration, die openUTM als erste SYSLOG öffnen soll, den Linknamen SYSLOG mit dem folgenden BS2000-Kommando zuweisen:

```
/SET-FILE-LINK LINK-NAME=SYSLOG,FILE-NAME=fgg-name(*gen)
```

gen ist die Generationsnummer der Dateigeneration, die openUTM nach dem Anwendungsstart als erste SYSLOG-Datei öffnen soll.

4.2.2.2 Dateigeneration anlegen

Vor dem Öffnen einer Dateigeneration überprüft openUTM, ob die entsprechende Dateigeneration schon existiert. Existiert die Dateigeneration nicht, dann legt openUTM die Dateigeneration an. Sie können die Dateigeneration jedoch auch selbst anlegen, z.B. um andere Werte für PRIMARY-ALLOCATION und SECONDARY-ALLOCATION zu definieren.

Anlegen der Dateigeneration durch openUTM

Abhängig davon, ob die FGG auf PUBLIC oder PRIVATE DISK angelegt wurde, setzt openUTM dazu intern eines der folgenden Kommandos ab:

- FGG auf PUBLIC DISK

```
/CREATE-FILE-GENERATION                        -
/   GENERATION-NAME = fgg-name(*gen)           -
/  ,SUPPORT = *PUBLIC-DISK(                    -
/      SPACE = RELATIVE(                       -
/         PRIMARY-ALLOCATION = prim-alloc      -
/        ,SECONDARY-ALLOCATION = sec-alloc ) )
```

- FGG auf PRIVATE DISK

```
/CREATE-FILE-GENERATION                       -
/   GENERATION-NAME = fgg-name(*gen)          -
/  ,SUPPORT = *PRIVATE-DISK(                  -
/      VOLUME = volume                        -
/     ,DEVICE-TYPE = device                   -
/     ,SPACE = RELATIVE(                      -
/         PRIMARY-ALLOCATION = prim-alloc     -
/        ,SECONDARY-ALLOCATION = sec-alloc ) )
```

Bedeutung der Parameter:

fgg-name Name der an openUTM übergebenen Dateigenerationsgruppe

gen Generation, die gerade zu öffnen ist

PUBLIC-DISK oder PRIVATE-DISK

gibt an, ob die Dateigeneration auf PUBLIC-DISK oder PRIVATE-DISK angelegt werden soll. Diese Information entnimmt openUTM dem Katalogeintrag der übergebenen SYSLOG-FGG.

VOLUME=volume, DEVICE=device

Datenträgerkennzeichen und Gerätetyp der Platte, auf der die FGG eingerichtet werden soll.

PRIMARY-ALLOCATION=prim-alloc, SECONDARY-ALLOCATION=sec-alloc

Größe der Anfangszuweisung von Speicherplatz bzw. Größe der Speicherplatzerweiterung.

Legt openUTM alle Generationen der FGG selbst an, dann setzt openUTM für beide Parameter 192 PAM-Seiten ein. Hat openUTM vor dem Anlegen der Dateigeneration bereits eine vorhandene (von Ihnen angelegte) Dateigeneration dieser FGG geöffnet, dann übernimmt openUTM deren Werte für Primary und Secondary Allocation beim Anlegen weiterer Dateigenerationen.

Dateigeneration selbst anlegen

Wollen Sie die Werte für Primary und Secondary Allocation für alle Dateigenerationen der SYSLOG-FGG selbst festlegen, dann müssen Sie also mindestens eine Dateigeneration mit den gewünschten Speicherplatz-Angaben selbst anlegen und diese Dateigeneration openUTM als erste SYSLOG-Datei übergeben. Alle weiteren Dateigenerationen werden von openUTM dann automatisch mit den von Ihnen verwendeten Werten für Primary und Secondary Allocation angelegt.

Die Dateigeneration können Sie mit dem folgenden BS2000-Kommando einrichten:

```
/CREATE-FILE-GENERATION                          -
/   GENERATION-NAME = filebase.SLOG(*1)          -
/  ,SUPPORT = *PUBLIC-DISC(                      -
/      SPACE = RELATIVE(                         -
/         PRIMARY-ALLOCATION = prim-alloc        -
/        ,SECONDARY-ALLOCATION = sec-alloc ) )
```

Liegt die FGG auf einer Privatplatte, dann müssen Sie die Dateigeneration auch auf Privatplatte anlegen.

Dateigeneration an openUTM übergeben

Übergeben Sie dann diese Dateigeneration als erste SYSLOG-Datei an openUTM. Dies kann entweder durch Verwenden des Linknamens SYSLOG oder durch Festlegen der Basis der FGG (siehe "Dateigenerationsgruppe SYSLOG-FGG") geschehen.

Wenn Sie mit der zweiten Methode arbeiten, müssen Sie beachten, dass openUTM die Basis nicht verändert. Falls also die Basis der FGG im gültigen Bereich der FGG liegt und falls Sie die Basis vor dem nächsten Start der Anwendung nicht auf eine andere Generation legen, beginnt openUTM auch beim nächsten Anwendungsstart mit derselben
(Basis-)Dateigeneration als erste SYSLOG-Datei. Die Informationen aus dem vorherigen Anwendungslauf gehen dann u.U. verloren. Darüber hinaus bleiben unabhängig davon, was Sie beim Einrichten der FGG für OVERFLOW-OPTION angegeben haben, nur die *n* jüngsten Dateigenerationen erhalten (*n*=MAXIMUM in CREATE-FILE-GROUP). Beachten Sie auch den Abschnitt „SYSLOG-Generationen erhalten“.

Damit openUTM auch hier nach Anwendungsende und erneutem Start mit der nächsten Generation als erste SYSLOG anfängt, sollten Sie vor jedem Start folgende Schritte ausführen:

1. Die Basis auf die zuletzt beschriebene Generation der FGG (LAST-GEN) setzen

```
/MODIFY-FILE-GROUP-ATTRIBUTES                    -
/   GROUP-NAME = filebase.SLOG                   -
/  ,GENERATION-PARAMETER = *GENERATION-PARAMETER(-
/      BASE-NUMBER = RELATIVE-TO-LAST-GEN( 0 ) )
```

2. Die nächste Dateigeneration anlegen

```
/CREATE-FILE-GENERATION                -
/   GENERATION-NAME = filebase.SLOG(+1) -
/   ......
```

3. Die Basis auf die gerade angelegte Dateigeneration setzen

```
/MODIFY-FILE-GROUP-ATTRIBUTES                     -
/   GROUP-NAME = filebase.SLOG                    -
/  ,GENERATION-PARAMETER = *GENERATION-PARAMETER( -
/      BASE-NUMBER = *RELATIVE-TO-LAST-GEN( 0 ) )
```

Diese Kommandofolge dürfen Sie nur einmal pro Anwendungsstart eingeben, nicht für jede gestartete UTM-Task.

4.2.2.3 Benutzerkennungs-Überlaufschutz

Sie kontrollieren die Belegung von Speicherplatz durch die SYSLOG-FGG wie folgt:

1. Schalten Sie die automatische Größenüberwachung ein, entweder bei der UTM-Generierung mit MAX..., SYSLOG-SIZE=*size*, oder über die Administration (z.B. mit dem Administrationskommando KDCSLOG SIZE= *size)*. Für *size* muss in beiden Fällen ein Wert > 0 angegeben werden.
2. Richten Sie die SYSLOG-FGG mit dem folgenden Kommando ein:

```
/CREATE-FILE-GROUP                                          -
/   GROUP-NAME = filebase.SLOG                              -
/  ,GENERATION-PARAMETER = *GENERATION-PARAMETER(           -
/      MAXIMUM = n                                          -
/ ,OVERFLOW-OPTION = CYCLIC-REPLACE oder REUSE-VOLUME )
```

Die Dateigenerationen werden zyklisch überschrieben, so dass in der FGG maximal *n* Generationen katalogisiert sind. Zusätzlich kann es vorkommen, dass auf dem BS2000-System intern noch eine weitere (ältere) Dateigeneration Speicherplatz belegt. Jede Generation hat durch die Größenüberwachung eine maximale Größe von *size* UTM-Seiten.

Damit ist der maximale Speicherverbrauch der SYSLOG-FGG kleiner oder gleich: (*n*+1) * *size* * (Größe einer UTM-Seite).

4.2.2.4 SYSLOG-Generationen erhalten

Wollen Sie alle Dateigenerationen der SYSLOG-FGG behalten, dann müssen Sie die SYSLOG-FGG wie folgt einrichten:

```
/CREATE-FILE-GROUP                                  -
/   GROUP-NAME = fgg-name                           -
/  ,GENERATION-PARAMETER = *GENERATION-PARAMETER(   -
/      MAXIMUM = n                                  -
/     ,OVERFLOW-OPTION = *KEEP-GENERATION )
```

In diesem Fall steht die Basis auf 0, d.h. sie liegt außerhalb des gültigen Bereichs zwischen erster und letzter Dateigeneration. Das BS2000 behält dann alle Generationen der FGG unabhängig von der Angabe im Parameter MAXIMUM (max. Anzahl der Generationen).

> **ACHTUNG!**
>
> Legen Sie die Basis der FGG jedoch in den gültigen Bereich, also zwischen erster und letzter Generation, dann gehen (ohne Warnung) alle Generationen, die außerhalb des gültigen Bereichs liegen verloren (Generationsnummern außerhalb von LAST-GEN - MAXIMUM und LAST-GEN).

Liegt die Basis innerhalb des gültigen Bereichs der FGG und wollen Sie trotzdem möglichst viele Generationen erhalten, dann müssen Sie *n* genügend groß (am besten 9999) wählen. In diesem Fall können Sie für den Operanden OVERFLOW-OPTION auch REUSE-VOLUME oder CYCLIC-REPLACE angeben.

4.2.2.5 Automatische Größenüberwachung

Die automatische Größenüberwachung können Sie nur für FGGs verwenden. Wenn Sie die SYSLOG-Datei als einfache Datei anlegen und die automatische Größenüberwachung generieren, dann bricht openUTM den Start der Anwendung mit Startfehlercode 58 ab.

Sie können die automatische Größenüberwachung auf zwei Arten einstellen:

- bei der UTM-Generierung mit der KDCDEF-Anweisung MAX ...,SYSLOG-SIZE=*size*
- im laufenden Betrieb der Anwendung mit dem Administrationskommando KDCSLOG [SWITCH,]SIZE=*size* oder an der Programmschnittstelle zur Administration mit dem Operationscode KC_SYSLOG und Subopcode KC_CHANGE_SIZE (siehe openUTM-Handbuch „Anwendungen administrieren“)

In beiden Fällen müssen Sie für *size* einen Wert > 0 angeben.

Bei eingeschalteter Größenüberwachung überprüft openUTM vor jeder Meldungsausgabe in die SYSLOG-Datei, ob durch die Meldungsausgabe die vereinbarte Maximalgröße der Dateigeneration (*size* * Größe einer UTM-Seite) überschritten würde. Ist dies der Fall, dann wird vor der Meldungsausgabe versucht, auf die nächste Dateigeneration zu schalten. Bei Erfolg gibt openUTM die Meldung K137 aus.

Tritt bei dem Versuch umzuschalten ein Fehler auf, dann arbeitet openUTM mit der alten Dateigeneration weiter, in die vor dem Umschaltversuch protokolliert wurde. openUTM schreibt die Meldung K139 auf SYSOUT und auf die Konsole des Systemoperateurs. Außerdem wird wie bei allen DMS-Fehlern zusätzlich die Meldung K043 ausgegeben. Sie enthält einen DMS-Fehlercode, dem Sie den Grund für den Umschaltfehler entnehmen können.

Damit openUTM nicht bei jeder folgenden Meldung mit dem Ziel SYSLOG erneut erfolglos versucht, auf die nächste Dateigeneration umzuschalten, wird die automatische Größenüberwachung nach einem solchen Umschaltfehler deaktiviert.

Nachdem der Administrator den Grund für den Umschaltfehler gefunden und beseitigt hat, kann er die automatische Größenüberwachung wieder aktivieren, z.B. mit dem Kommando KDCSLOG SWITCH. Mit dem KDCSLOG SWITCH zwingt er openUTM, einen erneuten Umschaltversuch zu starten. Verläuft dieser Versuch ohne Fehler, so wird eine vorher deaktivierte Größenüberwachung automatisch wieder aktiviert.

openUTM garantiert, dass nach dem Umschalten keine Meldung mehr in die alte Dateigeneration geschrieben wird. Das heißt jedoch nicht, dass Sie sofort frei über die alte Dateigeneration verfügen können. Erst müssen alle Tasks der Anwendung diese Dateigeneration schließen. Dies kann eventuell länger dauern, wenn Tasks sehr lange in Anwender-Teilprogrammen beschäftigt sind. Wird eine Dateigeneration von der letzten Task geschlossen, so gibt openUTM die Meldung K138 aus.

Welche Dateigenerationen geschlossen sind, können Sie der Ausgabe des Administrationskommandos KDCSLOG INFO entnehmen (LOWEST-OPEN-GEN).

openUTM verändert die vom Benutzer zu Anwendungsbeginn eingestellte FGG-Basis nicht. Dadurch gehen auch bei eingestellter FGG-Option OVERFLOW-OPTION= KEEP-GENERATION nicht ungewollt Dateigenerationen verloren.

4.2.3 Verhalten bei Schreibfehlern

Tritt bei dem Versuch, eine Meldung in die SYSLOG zu schreiben, ein Fehler auf, dann gibt openUTM die Meldung K043 aus, die einen DMS-Fehlercode enthält. An diesem Fehlercode können Sie den Grund für den Fehler ablesen.

Das weitere Vorgehen von openUTM ist abhängig davon, ob die SYSLOG als einfache Datei oder als FGG geführt wird.

- Die SYSLOG wird als einfache Datei geführt

  Nach Ausgabe der Meldung K043 wird die Anwendung mit Grund SLOG09 abgebrochen.

- Die SYSLOG wird als FGG geführt

  openUTM versucht beim Auftreten eines Fehlers auf die nächste Dateigeneration zu schalten. openUTM schaltet auch um, wenn die Größenüberwachung ausgeschaltet bzw. nicht generiert ist. openUTM schaltet nicht um, wenn die Größenüberwachung auf Grund eines vorangegangenen Umschaltfehlers suspendiert ist.

  Schlägt der Umschaltversuch fehl, dann wird die Anwendung mit Grund SLOG09 abgebrochen.

  Kann openUTM auf die nächste Dateigeneration umschalten, dann versucht openUTM erneut die Meldung in die SYSLOG zu schreiben. Tritt dabei ein Fehler auf, wird die Anwendung mit SLOG09 abgebrochen. Tritt kein Fehler auf, läuft die Anwendung weiter, openUTM protokolliert in die neue SYSLOG-Dateigeneration.

4.3 Benutzer-Protokolldatei

In der Benutzer-Protokolldatei stehen die Sätze, die das Anwendungsprogramm mit LPUT-Aufrufen erzeugt hat. Die Benutzer-Protokolldatei muss als Dateigenerationsgruppe eingerichtet werden.

Die Benutzer-Protokoll-Sätze schreibt openUTM nicht sofort in die Protokolldatei, sondern speichert sie erst im Pagepool der KDCFILE. Sind im Pagepool so viele UTM-Seiten durch LPUT-Sätze belegt, wie in MAX...,LPUTBUF=*number* generiert, kopiert openUTM die Sätze in die Benutzer-Protokolldatei. Das Kopieren erfolgt ausserhalb der Benutzer-Transaktion. Beim normalen Beenden der Anwendung kopiert openUTM die Sätze ebenfalls in die Benutzer-Protokolldatei.

Die Anzahl der bei LPUTBUF=*number* angegebenen UTM-Seiten ist bei der UTM-Generierung der Größe des Pagepools mit MAX...,PGPOOL=*number* zu berücksichtigen.

Mit MAX...,LPUTLTH=*length* beeinflussen Sie die Blocklänge der Benutzer-Protokolldatei. Sie wird von openUTM berechnet und kann größer sein als die Standardblockung 2KB.

openUTM kann LPUT-Sätze nur in die Benutzer-Protokolldatei kopieren, wenn diese eingerichtet ist und openUTM darauf zugreifen kann.

Bitte beachten Sie, dass die Benutzer-Protokolldatei nach einem KDCDEF-Lauf oder nach einem KDCUPD-Lauf von Anfang an überschrieben wird; ansonsten wird sie fortgeschrieben. Deshalb sollten Sie die Protokoll-Sätze vor einem KDCDEF- bzw. KDCUPD-Lauf auswerten oder die Basis der Benutzerprotokoll-Datei auf eine neue Dateigeneration legen.

4.3.1 Benutzer-Protokolldatei einrichten

Die Dateigenerationsgruppe für die Benutzer-Protokolldatei muss den Dateinamen *filebase*.USLA haben. *filebase* ist der in der MAX-Anweisung generierte Basisname der KDCFILE.

Die Dateigenerationsgruppe müssen Sie vor dem ersten Starten der Anwendung einrichten. Dazu sind folgende BS2000-Kommandos notwendig:

- CREATE-FILE-GROUP-Kommando zum Erstellen des Katalogeintrags. Sie müssen angeben:
  - den Namen der Dateigenerationsgruppe,
  - die maximal erlaubte Anzahl von Generationen,
  - die Vorgehensweise, wenn die maximale Anzahl Generationen erreicht ist (OVER-FLOW-OPTION=CYCLIC-REPLACE ist Standard)
  - evtl. die Zugriffsberechtigung.
- CREATE-FILE-GENERATION-Kommando zum Erstellen wenigstens einer Generation.
- MODIFY-FILE-GROUP-ATTRIBUTES-Kommando zum Festlegen der Bezugsgeneration für relative Nummerierung.

Beispiel: Dateigenerationsgruppe für eine Benutzer-Protokolldatei einrichten

```
/CREATE-FILE-GROUP GROUP-NAME=filebase.USLA                    -
/                 ,GENERATION-PARAMETER=GENERATION-PARAMETER(  -
/                  MAXIMUM=3,OVERFLOW-OPTION=REUSE-VOLUME)
/CREATE-FILE-GENERATION GENERATION-NAME=filebase.USLA(+1)
/MODIFY-FILE-GROUP-ATTRIBUTES GROUP-NAME=filebase.USLA         -
/                 ,GENERATION-PARAMETER=GENERATION-PARAMETER(  -
/                  BASE-NUMBER=RELATIVE-TO-LAST-GEN(NUMBER=0))
```

Der Parameter OVERFLOW-OPTION=REUSE-VOLUME bewirkt, dass eine neue Generation auf demselben Datenträger eingerichtet wird wie die dafür gelöschte.

Falls die Benutzerprotokoll-Sätze (LPUT-Aufrufe) > 6 KB werden können, müssen Sie in den CREATE-FILE-GENERATION-Kommandos den SPACE-Operanden für die Primär- und Sekundärzuweisung angeben. Die Werte des SPACE-Operanden müssen so groß sein, dass mindestens ein LPUT-Satz in den durch die Sekundärzuweisung belegten Plattenbereich passt. Die Primärzuweisung muss mindestens doppelt so groß sein wie die Sekundärzuweisung.

Soll die Dateigenerationsgruppe auf einer Privatplatte eingerichtet werden, müssen Sie Folgendes zusätzlich beachten:

- Jede einzelne Dateigeneration der Gruppe müssen Sie vor dem Start der Anwendung mit einem CREATE-FILE-GENERATION-Kommando auf der Privatplatte einrichten.
- Sie müssen beim Einrichten der Dateigeneration OVERFLOW=REUSE-VOLUME angeben.
- Wird die Basis der Dateigenerationen nach dem Einrichten auf das letzte Element der Dateigenerationsgruppe gesetzt, beginnt openUTM mit der letzten Dateigeneration und schaltet danach auf die nächste Generation weiter.
  Wird als Basis die erste Generation angegeben, dann schaltet openUTM beim ersten Kommando KDCLOG (siehe "Umschalten auf die nächste Dateigeneration") auf die letzte Generation weiter.

Bei Dateigenerationen auf PUBLIC-DISK entfallen diese Einschränkungen.

Beispiel: Dateigenerationsgruppe auf Privatplatte einrichten

```
/CREATE-FILE-GROUP GROUP-NAME=filebase.USLA                        -
/                  ,GENERATION-PARAMETER=GENERATION-PARAMETER(     -
/                   MAXIMUM=3,OVERFLOW-OPTION=REUSE-VOLUME         -
/                  ,VOLUME=B004H,DEVICE-TYPE=D3465)
/CREATE-FILE-GENERATION GENERATION-NAME=filebase.USLA(*1)           -
/                  ,SUPPORT=PRIVATE-DISK(VOLUME=B004H,DEVICE-TYPE=D3435)
/CREATE-FILE-GENERATION GENERATION-NAME=filebase.USLA(*2)           -
/                  ,SUPPORT=PRIVATE-DISK(VOLUME=B004H,DEVICE-TYPE=D3435)
/CREATE-FILE-GENERATION GENERATION-NAME=filebase.USLA(*3)           -
/                  ,SUPPORT=PRIVATE-DISK(VOLUME=B004H,DEVICE-TYPE=D3435)
/MODIFY-FILE-GROUP-ATTRIBUTES GROUP-NAME=filebase.USLA              -
/                  ,GENERATION-PARAMETER=GENERATION-PARAMETER(     -
/                   BASE-NUMBER=RELATIVE-TO-LAST-GEN(NUMBER=0))
```

4.3.2 Doppelte Benutzer-Protokolldatei

Wird mit doppelten Benutzer-Protokolldateien gearbeitet (MAX...,USLOG=DOUBLE), so ist eine zweite Dateigenerationsgruppe mit dem Namen *filebase*.USLB auf gleiche Weise einzurichten wie die erste Dateigenerationsgruppe *filebase*.USLA. Die Dateigenerationsgruppe *filebase*.USLB muss auf demselben Katalog liegen wie die KDCFILE-Teile mit dem Suffix B.

4.3.3 Umschalten auf die nächste Dateigeneration

Zum Weiterschalten auf die nächste Dateigeneration stehen dem Administrator der Anwendung das Kommando KDCLOG und an der Programmschnittstelle KDCADMI der Opcode KC_USLOG zur Verfügung. Bei jedem Absetzen des KDCLOG-Kommandos bzw. eines entsprechenden KDCADMI-Aufrufs schaltet openUTM auf die jeweils folgende Dateigeneration weiter.

Beim ersten Start der UTM-Anwendung schreibt openUTM die Benutzer-Protokollsätze in die Generation der Gruppe, die beim Einrichten als Basis angegeben wurde. Bei jedem Weiterschalten durch den Administrator setzt openUTM ein MODIFY-FILE-GROUP-ATTRIBUTES-Kommando auf die nächste Dateigeneration ab und setzt die Basis auf die dann jüngste Dateigeneration.

Weiterschalten auf die nächste Generation können Sie auch wie folgt nach normaler Beendigung der UTM-Anwendung:

```
/CREATE-FILE-GENERATION GENERATION-NAME=filebase.USLA(+1)
/MODIFY-FILE-GROUP-ATTRIBUTES GROUP-NAME=filebase.USLA,                    -
/                             GENERATION-PARAMETER=GENERATION-PARAMETER(   -
/                             BASE-NUMBER=RELATIVE-TO-LAST-GEN(NUMBER=0))
```

Diese Kommandos zum Weiterschalten auf die nächste Dateigeneration können Sie auch in die Startprozedur Ihrer UTM-Anwendung übernehmen.

Will der UTM-Administrator z.B. mit KDCLOG auf eine andere Benutzer-Protokolldatei umschalten, dann muss die Dateigenerationsgruppe unter der gleichen BS2000-Benutzerkennung eingerichtet sein, unter der die UTM-Prozesse laufen.

4.3.4 Verhalten bei Schreibfehlern

Tritt beim Schreiben von LPUT-Sätzen in die Benutzer-Protokolldatei ein DMS-Fehler (**D**ata **M**anagement **S**ystem) auf, so gibt openUTM die Meldung K043 aus, die einen DMS-Fehlercode enthält. An diesem Fehlercode können Sie den Grund für den Fehler ablesen. Gleichzeitig wird jeder weitere LPUT-Aufruf im Teilprogramm mit dem KDCS-Returncode 40Z (interner Returncode K903) abgewiesen.

Der Administrator der Anwendung kann dann die Benutzer-Protokolldatei bzw. ihre Generationen korrigieren, restaurieren oder neu einrichten.

Damit openUTM wieder LPUT-Sätze in die Benutzer-Protokolldatei schreibt, muss der Administrator der Anwendung nach der Fehlerbehebung das Administrationskommando KDCLOG oder einen KDCADMI-Aufruf mit Opcode KC_USLOG absetzen (siehe openUTM-Handbuch „Anwendungen administrieren“).

Die Dateigenerationsnummer wird erhöht. Die im Pagepool der KDCFILE gesicherten LPUT-Sätze werden anschließend in die Protokolldatei(en) geschrieben. Die Sperre für die LPUT-Aufrufe in den Teilprogrammen wird aufgehoben.

5 UTM-Anwendung starten

Eine UTM-Anwendung kann als ENTER-Task von der BS2000-Konsole oder von jedem Terminal gestartet werden.

Unter der dabei verwendeten BS2000-Benutzerkennung wird die CPU-Zeit usw. für die Anwendung abgerechnet.

Alle nötigen Dateien müssen entweder unter dieser Kennung katalogisiert sein oder mit SHARE=YES unter einer anderen Kennung auf demselben Host.

Benötigt werden die folgenden Dateien:

- Bibliothek mit einem Start-LLM
- KDCFILE
- Benutzer-Protokolldatei (USLOG, optional)
- System-Protokolldatei (SYSLOG)
- Benutzerdateien der Anwendung
- Modulbibliothek SYSLNK.UTM.070
- Bibliotheken mit den Verbindungsmodulen für die Datenbanksysteme (optional)
- Bibliothek mit dem Verbindungsmodul für das Formatierungssystem (optional)
- Bibliothek mit den Formaten (optional)

Die ENTER- oder Prozedur-Datei zum Starten der UTM-Anwendung enthält mindestens:

- CREATE-FILE- und SET-FILE-LINK-Kommando für System-Protokolldatei
- CREATE-FILE- und SET-FILE-LINK-Kommandos für Benutzerdateien (optional)
- START-EXECUTABLE-PROGRAM-Kommando zum Aufruf des Anwendungsprogrammes
- Startparameter für UTM, FHS und Datenbanksysteme

Wenn eine ENTER-Datei erstellt wurde, wird die Anwendung durch den Aufruf des BS2000-Kommandos ENTER-JOB gestartet:

```
/ENTER-JOB FROM-FILE=enterfile[,JOB-CLASS=job-class]  -
/ [,RESOURCES=PARAMETERS(CPU-LIMIT=tttt)]
```

Im Falle einer SDF-Start-Prozedur-Datei verwenden Sie zum Starten der Anwendung das BS2000-Kommando ENTER-PROCEDURE:

`/ENTER-PROCEDURE FROM-FILE=enter-proc-file` (mit gleichen Parametern)

Empfehlungen für die Wahl der Parameter:

- Für den ENTER-Job einer UTM-Anwendung sollten Sie eine eigene Jobklasse einrichten, in der Sie die wichtigen Parameter des ENTER-Jobs einstellen.

  Die Jobklasse können Sie dann im ENTER-PROC-Kommando oder im ENTER-JOB-Kommando (bzw. bei den Parametern zu SET-LOGON-PARAMETERS in einer ENTER-JOB-Datei) zuweisen.
- Der Operand CPU-LIMIT (CPU-Zeitbeschränkung) sollte im ENTER-PROC oder im ENTER-JOB-Kommando angegeben werden, falls er nicht bei einer ENTER-JOB-Datei im Kommando SET-LOGON-PARAMETERS angegeben ist.

  Sie sollten den Wert so einstellen, dass die Jobs einer UTM-Anwendung keiner CPU-Zeitbeschränkung unterliegen. Aus diesem Grund sollten Sie CPU-LIMIT=NO (bzw. TIME=NTL) setzen bzw. die Jobklasse entsprechend definieren.

  Wird die CPU-Zeit beschränkt, d.h. CPU-LIMIT *ungleich* NO gesetzt, und tritt für eine Task der Anwendung ein CPU-Time-Runout auf, dann kann dies zum abnormalen Ende der UTM-Anwendung führen!

  Bitte beachten Sie, dass für den Ablauf der Enter-Tasks ohne CPU-Zeitbegrenzung eventuell eine NTL-Berechtigung (**N**o **T**ime **L**imit) im Benutzereintrag für die betreffende Abrechnungsnummer notwendig ist.

  Eine Beschränkung des CPU-Verbrauchs von einzelnen Teilprogrammen einer UTM-Anwendung können Sie bei der KDCDEF-Generierung im Operanden TIME der TAC-Anweisung einstellen.

Beim Start der UTM-Anwendung werden alle Voraussetzungen für den Betrieb der Anwendung geschaffen, d.h. Bereiche und Tabellen eingerichtet, Dateien geöffnet, Verbindungen aufgebaut etc. Bei diesen Aktionen können Fehlersituationen auftreten, welche die Startroutine erkennt und die evtl. zum Abbruch des Starts der Anwendung oder einer Task führen. openUTM gibt dann die Meldung K078 oder K049 nach SYSOUT aus, die in einem Fehlercode die Ursache des Abbruchs anzeigt (siehe hierzu openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“).

In der Startprozedur sind keine Kommandos zur Konsistenzprüfung auf Dateien notwendig, die von openUTM geöffnet werden (REPAIR-DISK-FILES bzw. CHECK-FILE-CONSISTENCY). openUTM ruft beim Öffnen von Dateien, die nicht ordnungsgemäß geschlossen wurden, selbst diese Kommandos auf.

Das Anwendungsprogramm bleibt solange geladen, bis es entweder durch Eingriff des Administrators oder infolge eines Fehlers beendet wird. Wie openUTM das Anwendungsprogramm beendet, ist wichtig für den Aufbau der Startprozedur durch den Anwender.

Es gibt folgende Fälle:

- Bei einer normalen administrativen Beendigung (z.B. über WinAdmin/WebAdmin oder das Administrationskommando KDCSHUT NORMAL/WARN/GRACE) oder nach einem Abbruch der Anwendung (Term Application) beendet openUTM das Anwendungsprogramm mit TERM UNIT=STEP, d.h. in der Start-Prozedur werden alle Kommandos bis zum nächsten /SET-JOB-STEP bzw. bis /EXIT-JOB bzw. /LOGOFF ignoriert bzw. bei der Verwendung einer SDF-Prozedur bis zum nächsten IF-BLOCK-ERROR Kommando übersprungen..
- Beim Austausch des Anwendungsprogrammes mit KDCAPPL PROG=NEW oder nach einem Programmfehler, der zum PEND ER führt, soll das Anwendungsprogramm neu geladen und gestartet werden. openUTM beendet das Programm dann mit TERM UNIT=PRGR, d.h. das nächste Kommando der Start-Prozedur wird interpretiert. Dort sollte ein Kommando /SKIP-COMMANDS stehen, das zum Kommando /START-EXECUTABLE-PROGRAM für den Start des Anwendungsprogrammes zurückführt.

Der Anwender kann die Verarbeitung in der Startprozedur nach dem Kommando /SET-JOB-STEP bzw. IF-BLOCK-ERROR davon abhängig machen, ob die UTM-Anwendung normal oder abnormal beendet wurde:

Bei einer abnormalen Beendigung der Anwendung (Term Application) erzeugt openUTM in der Task File Table (TFT) einen Eintrag mit LINK-NAME=KDCTRMAP. Man kann in der Startprozedur (mit /SHOW-FILE-LINK) abfragen, ob ein solcher Eintrag in der TFT vorhanden ist und abhängig davon den weiteren Ablauf steuern.

Beim Schreiben eines UTM-Dumps wird der Linkname KDCDUMP für die Dump-Datei vergeben. Ist dieser Link-Name also vorhanden, kann die Dump-Datei schon in der Startprozedur mit Hilfe des UTM-Tools KDCDUMP aufbereitet werden.
Eine andere Möglichkeit, den weiteren Prozedurlauf nach einem abnormalen Anwendungsende zu steuern und z.B. die Anwendung erneut zu starten,, bietet die Überwachung mit Jobvariablen (siehe Abschnitt „Nach abnormalem Anwendungsende erneut starten“).

Die Startprozedur ist in den folgenden Abschnitten beschrieben. Sie steht in einer Datei, die in diesem Kapitel mit *enterfile* oder *enter-proc-file* bezeichnet wird.

Mit der Startprozedur erzeugen Sie eine ENTER-Task. Die benutzte Kennung ist im ENTER-JOB- bzw. ENTER-PROC-Kommando oder im SET-LOGON-PARAMETERS-Kommando der Startprozedur anzugeben.

Mit ENTER-JOB bzw. ENTER-PROC gestartete UTM-Tasks sind Batch-Tasks. Damit unterliegen sie den JOB-CLASS-Einschränkungen für Batch-Tasks. Im Normalfall will man jedoch UTM-Tasks sofort starten. Das lässt sich durch die Funktion „JOB EXPRESS“ erreichen. Dazu muss im Benutzereintrag der BS2000-Benutzerkennung, unter der die Tasks ablaufen sollen, START-IMMEDIATE=YES eingetragen sein.

In BS2000-Systemen gibt es neben den Batch- und Dialogtasks die Klasse der TP-Tasks. Sie behandelt das Betriebssystem bevorzugt. openUTM meldet die mit ENTER-JOB bzw. ENTER-PROC gestarteten Tasks als TP-Tasks an. Diese Tasks werden aber nur dann bevorzugt behandelt, wenn unter der entsprechenden BS2000-Benutzerkennung bzw. Job-Klasse TP-Tasks erlaubt sind. Sind TP-Tasks verboten, laufen UTM-Tasks als Batch-Tasks.

Alle Tasks einer Anwendung müssen unter der gleichen BS2000-Benutzerkennung gestartet werden.

i
- Eine UTM-Anwendung kann auch im Dialog gestartet werden. Dies sollte aber nur zu Testzwecken geschehen, siehe auch openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“.
- Die Startprozedur kann auch durch WinAdmin/WebAdmin aufgerufen werden. Dazu muss neben WinAdmin/WebAdmin auch das Produkt openFT installiert sein (auf dem BS2000-System und auf dem WinAdmin- bzw. WebAdmin-Rechner).

- Eine Beispiel-Startprozedur entnehmen Sie dem Abschnitt „Grundstruktur für eine SDF-Startprozedur“

5.1 Startparameter der Anwendung

Beim Start des Anwendungsprogramms liest die Main Routine KDCROOT Startparameter von SYSDTA ein.

Mit den Startparametern werden aktuelle Ablaufparameter der Anwendung festgelegt. Dazu gehören z.B. die Anzahl der Tasks, mit der die Anwendung gestartet werden soll, oder Parameter für konfigurierte Datenbank- und /oder Formatierungssysteme.

Die Startparameter müssen in zwei Blöcken angegeben werden:

- 1. Block: Startparameter für openUTM
- 2. Block: Startparameter für Datenbank- und Formatierungssystem

Innerhalb eines Blocks ist die Reihenfolge der Startparameter beliebig. Jeder Block - auch ein leerer Block - wird mit einem END-Kommando abgeschlossen. Wenn kein Datenbank- und kein Formatierungssystem generiert ist, folgen zwei END-Kommandos aufeinander.

UTM-Startparameter im zweiten Block werden ignoriert. Das gleiche gilt für Startparameter des Datenbank- und Formatierungssystems im ersten Block.

Die Startparameter können in einer oder mehreren Zeilen angegeben werden. Ein Präfix legt fest, für wen die Startparameter bestimmt sind:

- Startparameter mit dem Präfix „.UTM“ oder ohne Präfix werden von openUTM selbst interpretiert.
- Startparameter mit dem Präfix „.UDS“, „.LEASY“, „.RMXA“, „.DB“ oder „.CIS“ leitet openUTM an das entsprechende Datenbanksystem zur Auswertung weiter.

  Beispiel

  Wenn die Anwendung z.B. mit DATABASE...,TYPE=UDS generiert wurde, ist als Präfix „.UDS“ für die Parameter anzugeben, die an UDS weitergereicht werden sollen.

  Wenn die Anwendung mit DATABASE ...,TYPE=XA generiert wurde, ist als Präfix „.RMXA“ anzugeben.
- Startparameter mit dem Präfix „.FHS" leitet openUTM an das Formatierungssystem FHS weiter, das mit dem Typ FHS in der FORMSYS-Anweisung generiert wurde.

5.1.1 Startparameter für openUTM

Die Syntax der UTM-Startparameter ist im Folgenden dargestellt.

```
[.UTM] START   FILEBASE=filebase  [ ,CATID=(catalog-A,catalog-B) ]

               [ ,ADMI-TRACE= { ON | OFF } ]
               [ ,ASYNTASKS=number ]
               [ ,BTRACE={{ ON | OFF } | ({ ON | OFF }, length )}]
               [, CPIC-TRACE = { TRACE | BUFFER | DUMP | ALL | OFF }]
               [ ,DB-CONNECT-TIME=sec ]

               [ ,DBKEY=db-key ]

               [ ,DUMP-CONTENT={ STANDARD | EXTENDED } ]
               [ ,DUMP-MESSAGE=(event-typ,event) ]

               [ ,DUMP-PREFIX=filename-prefix ]

               [ ,DUMP-USERID={ STANDARD | SYSUSER } ]
               [ ,ENTER-PROC-INPUT='enter-proc-file,[(par1=param1

                                  ,...,par<n>=param<n>),<enter-proc-pars>]'

               [ ,OTRACE={ ON | (SPI, INT, OSS, SERV, PROT) | OFF } ]

               [ ,PASSWORD=connection-password ]

               [ ,ROOTNAME=rootname ]
               [ ,STARTNAME={ enterfile |

                            'enterfile[,enteroperand][...]' } ]

               [ ,START-SSL-PROC=start-ssl-procedure,<enter-proc-pars>]

               [ ,STXIT={ ON | OFF } ]
               [ ,STXIT-LOG={ ON | OFF } ]

               [ ,SYSPROT=(interval,filename-prefix) ]

               [ ,TABLIB=libname ]
               [ ,TASKS=number ]

               [ ,TASKS-IN-PGWT=number ]

               [ ,TESTMODE={ ON | OFF | FILE } ]
               [ ,TX-TRACE = { ERROR | INTERFACE | FULL | DEBUG | OFF }]

               [ ,UTM-MSG-DATE={ YES | NO } ]

               [ ,XATMI-TRACE = { ERROR | INTERFACE | FULL | DEBUG | OFF }]

[.UTM] END
```

In obiger Syntax werden die Parameter in einer Zeile ohne Zeilenumbruch angegeben und jeweils durch ein Komma getrennt.

Sie können die Parameter beim START-Kommando jedoch auch auf mehrere Zeilen verteilen. In diesem Fall müssen Sie in jeder Zeile das Kommando START vor den Parameter stellen.

Syntax-Prüfung beim Start der Anwendung

- Erkennt openUTM bei der Überprüfung der Startparameter einen Syntaxfehler, dann gibt openUTM die Meldung K038 aus, setzt für den betroffenen Startparameter den Standardwert, sofern vorhanden, und startet die Anwendung.
- Bei einem Syntaxfehler im Parameter FILEBASE kann die Anwendung **nicht** gestartet werden, da kein Standardwert für diesen Parameter existiert.

Bedeutung der Kommandos

START Mit diesem Kommando gibt man die für den Lauf einer UTM-Anwendung erforderlichen UTM-Startparameter an. Die Anwendung wird nach der Eingabe aller Startparameter sofort gestartet.

END Dieses Kommando schließt die Eingabe der Startparameter ab (openUTM, Datenbank- und Formatierungssystem).

Die Startparameter müssen in eigenen Blöcken angegeben werden:

- zuerst die openUTM-Startparameter
- dann die Startparameter für die angeschlossenen Datenbank- und Formatierungssysteme.

Jeder Block wird mit dem END-Kommando abgeschlossen.

Bedeutung der Operanden

FILEBASE = filebase

Basisname für die KDCFILE und die Benutzer-Protokolldatei. Hier müssen Sie den Namen angeben, unter dem die KDCFILE zum Startzeitpunkt angelegt ist (max. 42 Zeichen). Bei Angabe eines ungültigen Namens wird der Start der Anwendung abgebrochen.

ADMI-TRACE =

Ein-/Ausschalten der ADMI-Tracefunktion (= Tracefunktion der Programmschnittstelle zur Administration KDCADMI), siehe auch openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“.

Zum Namen der Trace-Dateien siehe „Trace-Dateien“.

ON Beim Start der Anwendung wird die ADMI-Tracefunktion eingeschaltet.

OFF Beim Start der Anwendung bleibt die ADMI-Tracefunktion ausgeschaltet.

Standard: OFF

ASYNTASKS = number

Anzahl der Tasks, die max. für asynchrone Vorgänge arbeiten sollen.

Standard: in MAX...,ASYNTASKS=*number* festgelegte Anzahl

Minimalwert: 0

Maximalwert: in MAX...,ASYNTASKS=*number* festgelegte Anzahl

BTRACE =

Ein-/Ausschalten der BCAM-Tracefunktion.

ON

Beim Start der Anwendung wird die BCAM-Tracefunktion eingeschaltet.
Es werden alle Verbindungs-spezifischen Ereignisse in der BCAM-Tracedatei aufgezeichnet.

Damit der Trace geschrieben werden kann, müssen Sie in der UTM-Startprozedur für jede Task eine Trace-Datei einrichten und dieser mit dem Kommando SET-FILE-LINK den LINK-Name KDCBTRC zuweisen.

Näheres zum Einrichten der Trace-Datei, zum Auswerten mit dem UTM-Tool KDCBTRC und zur Beschreibung des Traces finden Sie im openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“.

OFF

Beim Start der Anwendung bleibt die BCAM-Tracefunktion ausgeschaltet.

Standard: OFF

length

gibt die maximale Länge der Daten an, die bei eingeschalteter BCAM-Tracefunktion aufgezeichnet werden. Wenn die aufzuzeichnenden Daten länger sind, werden die ersten *length*/2 Zeichen und die letzten *length*/2 Zeichen der Daten aufgezeichnet. Diese Länge kann nur über Startparameter festgelegt werden.

Standard: 256
Minimalwert: 32
Maximalwert: 32624

Wenn Sie den BCAM-Trace für die Funktion UPIC Capture einsetzen (siehe auch Abschnitt „UPIC-Conversation mitschneiden (UPIC Capture)“), dann wird empfohlen, den Maximalwert zu verwenden.

CATID = (catalog-A,catalog-B)

Mit diesem Parameter können Sie die Katalogkennungen angeben, denen die Teile der KDCFILE zugeordnet werden.

Wenn Sie mit CATID arbeiten, müssen Sie im Parameter FILEBASE den Basisnamen ohne CATID angeben. Bei doppelter Dateiführung können Sie die Dateien mit Suffix A *catalog-A* und die Dateien mit Suffix B *catalog-B* zuordnen. Arbeiten Sie mit doppelter Dateiführung und geben nur *catalog-A* an, werden beide Dateien derselben CATID zugeordnet.

CPIC-TRACE =

Ein-/Ausschalten der CPI-C-Tracefunktion (= Tracefunktion der X/Open-Schnittstelle CPI-C), siehe auch openUTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

Zum Namen der Trace-Dateien siehe „Trace-Dateien“ .

TRACE	Beim Start der Anwendung wird die CPI-C-Tracefunktion mit Level TRACE eingeschaltet. Zu jedem CPI-C-Funktionsaufruf wird der Inhalt der Input- und Output-Parameter ausgegeben. Von den Datenpuffern werden nur die ersten 16 Byte ausgegeben. Die Returncodes der KDCS-Aufrufe, auf die die CPI-C-Aufrufe abgebildet werden, werden ausgegeben.
BUFFER	Beim Start der Anwendung wird die CPI-C-Tracefunktion mit Level BUFFER eingeschaltet. Dieser Trace-Level beinhaltet den Level TRACE, die Datenpuffer werden jedoch in voller Länge protokolliert.
DUMP	Beim Start der Anwendung wird die CPI-C-Tracefunktion mit Level DUMP eingeschaltet. Dieser Trace-Level beinhaltet den Level TRACE, zusätzlich werden Diagnose-Informationen in die Trace-Datei geschrieben.
ALL	Beim Start der Anwendung wird die CPI-C-Tracefunktion mit Level ALL eingeschaltet. Dieser Trace-Level beinhaltet die Level BUFFER, DUMP und TRACE.
OFF	Beim Start der Anwendung bleibt die CPI-C-Tracefunktion ausgeschaltet.

DB-CONNECT-TIME = sec

Maximale Wartezeit in Sekunden für den Verbindungsaufbau zur Datenbank.

Erfolgt während der Wartezeit kein Verbindungsaufbau zur Datenbank, wird die Meldung K078 ausgegeben und die Task beendet.

Standard: 0 (keine Zeitüberwachung)Minimalwert: 60 Maximalwert: 3600

DBKEY = db-key

Bis zu acht Zeichen langer Schlüssel für den Zugriff zur Datenbank. Gilt nur für UDS-Datenbanken!

db-key wird bei Aufrufen an das Datenbanksystem verwendet, wenn für den Transaktionscode des aufrufenden Teilprogramms kein DBKEY definiert ist.

Standard: C’UTM '

DUMP-CONTENT =

gibt an, ob openUTM die Task-übergreifenden Speicherbereiche in allen Dumps einer Dump-Dateigeneration, d.h. für alle Tasks, abzieht oder nur im Dump der Task, die den Anwendungsabbruch verursacht hat.

STANDARD (STD)

Wenn openUTM eine Dump-Dateigeneration erzeugt, dann sind Task-übergreifende Speicherbereiche nur im Dump der ersten Task (Verursacher) enthalten. Für die Diagnose ist das normalerweise ausreichend.

Standard: STANDARD

EXTENDED (EXT)

Die Task-übergreifenden Speicherbereiche sind in allen Dumps einer DUMP-Dateigeneration enthalten.

> ¡ Diesen Wert sollten Sie nur auf besondere Anforderung des Service einstellen.

DUMP-MESSAGE = *(event-type,event)*

Ereignis, bei dem UTM bei eingeschaltetem Testmodus einen UTM-Dump erzeugt. Ein Dump wird nur von der Task erstellt, in der das Ereignis eingetreten ist; die Anwendung wird dabei nicht beendet.

Sie können für *event-type*,*event* Folgendes angeben:

event-type=MSG,*event*=K*nnn* (K-Meldung)

Der UTM-Dump wird erzeugt, wenn die Meldung K*nnn* auftritt. Bei den Meldungsnummern K023, K043, K061, K062 wird nur einmal ein Dump erzeugt, danach wird *event* automatisch zurückgesetzt.Bei allen anderen Meldungen wird solange bei jedem Auftreten der Meldungsnummer ein Dump erzeugt, bis der Wert per Administration zurückgesetzt wird.

Der Wert von DUMP-MESSAGE kann per Administration zurückgesetzt werden, z.B. durch WinAdmin/WebAdmin oder das Kommando KDCDIAG DUMP-MESSAGE=*NONE.

event-type=RCCC,event=rccc (Kompatibler KDCS-Returncode)

Für rccc geben Sie einen KDCS-Returncode (KCRCCC, z.B. 40Z) an. Beim Auftreten dieses Returncodes bei einem KDCS-Aufruf wird ein UTM-Dump mit Kennzeichen `CC-40Z` von der Task erzeugt, in der der Returncode aufgetreten ist. Anschließend wird der Message-Dump für dieses Ereignis automatisch ausgeschaltet.

event-type=RCDC,*event*=rcdc (interner KDCS-Returncode)

Für rcdc geben Sie einen inkompatiblen KDCS-Returncode (KCRCDC, z.B. KD10) an. Beim Auftreten dieses Returncodes bei einem KDCS-Aufruf wird ein UTM-Dump mit dem Kennzeichen DCKD10 von dem Prozess erzeugt, in dem der Returncode aufgetreten ist. Anschließend wird der Message-Dump für dieses Ereignis automatisch ausgeschaltet.

event-type=SIGN,*event*=sign (SIGN-Statuscode)

Für sign geben Sie einen SIGNON-Statuscode (KCRSIGN1/2, z.B. U05) an, wobei KCRSIGN1 den Wert U, I, A oder R haben muss. Beim Auftreten dieses Codes beim Anmelden eines Benutzers wird ein UTM-Dump mit Kennzeichen `SG-U05` von der Task erzeugt, bei der der SIGNON-Status aufgetreten ist. Das passiert unabhängig davon, ob in der Anwendung ein Anmelde-Vorgang generiert ist oder nicht. Anschließend wird der Message-Dump für dieses Ereignis automatisch ausgeschaltet.

Hinweise:
Bei allen KDCS-Returncodes >=70Z und den zugehörigen inkompatiblen KDCS-Returncodes, bei denen kein PENDER-Dump geschrieben wird (z.B. 70Z/K316), wird auch kein DUMP erzeugt.

Im Administrationskommando KDCDIAG oder durch die entsprechende Funktion bei WinAdmin/WebAdmin oder KDCADMI können mit den Parametern DUMP-MESSAGE1, DUMP-MESSAGE2 und DUMP-MESSAGE3 bis zu drei verschiedene Ereignisse angegeben werden, über den Startparameter *DUMP-MESSAGE* dagegen nur ein Ereignis. Außerdem können im Startparameter bei *event-type*=MSG keine Meldungs-Inserts angegeben werden, im Kommando KDCDIAG dagegen bis zu drei Inserts als zusätzliche Bedingungen.

Das Kennzeichen des Dumps ist abhängig vom Ereignis:

Ereignis	**Präfix**	**Beispiel**
K- oder P-Meldung	ME gefolgt von der Meldungsnummer	`MEP012`
Primärer KDCS-Returncode	CC- gefolgt vom Returncode	`CC-71Z`
Sekundärer KDCS-Returncode	DC gefolgt vom Returncode	`DCK303`
SIGN-Status	SG- gefolgt vom Status	`SG-U01`

DUMP-PREFIX

filename-prefix ist das Dateinamenspräfix für die UTM-Dump-Dateien. Es ist maximal 17 Zeichen lang.

Wenn Sie *filename-prefix* angeben, schreibt openUTM die Speicherabzüge in eine Dateigenerationsgruppe (FGG, File Generation Group) oder eine normale BS2000-Datei (Dump-Datei) mit Namen
filename-prefix`.rrrrrr.ttttff`.

Wenn Sie kein *filename-prefix* angeben, oder wenn der Speicherabzug (ohne Leerzeichen) in einer sehr frühen Phase beim Start der Anwendung geschrieben wird, schreibt openUTM die Speicherabzüge in eine Dateigenerationsgruppe oder eine Datei mit Namen
`DUMP.UTM.rrrrrr.ttttff.aaaaaaaa.`

`rrrrrr`	Returncode, der die Ursache der Beendigung angibt. Detaillierte Informationen finden Sie im openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“.
`tttt`	TSN der Task, die den Dump erzeugt hat
`ff`	Wenn die Dump-Datei keine FGG-Datei ist: Folgenummer für die Dumps, die ein Prozess einer Anwendung erzeugt. Wenn die Dump-Datei eine FGG-Datei ist: sedezimaler Wert der Zähler der Anwendungsstarts seit der KDCDEF-Generierung.
`aaaaaaaa`	Name der Anwendung, zu der der Dump gehört

DUMP-USERID =

gibt an, unter welcher Kennung die UTM-Dumps abgelegt werden sollen.

STANDARD (STD)	Die UTM-Dumps werden unter der Kennung abgelegt, unter der die UTM-Task gestartet wurde. Standard: STANDARD
SYSUSER (SYS)	Die UTM-Dumps werden unter der Kennung $SYSUSER abgelegt.

ENTER-PROC-INPUT = *enter-proc-input*

Angabe der ENTER-PROC-Datei und deren Parameter.
Wird dieser Parameter angegeben, so werden die Folgetasks einer UTM-Anwendung mittels eines ENTER-PROC-Kommandos gestartet.

Die Länge von *enter-proc-input* darf maximal 255 Zeichen betragen.

Dieser Parameter kann mehrfach angegeben werden, die Angabe muss jeweils in Hochkommata erfolgen. Die in mehreren ENTER-PROC-Anweisungen angegebenen Parameter werden in der Reihenfolge ihrer Angabe zusammengesetzt. Die maximale Länge der aufgesammelten Angaben beträgt 2000 Zeichen. Wird die maximale Gesamtlänge überschritten, so wird eine Meldung ausgegeben und der Start der Anwendung abgebrochen.

Die aufgesammelten *enter-proc-input*-Parameter werden von UTM mittels eines ENTER-PROC-Kommandos abgesetzt, sie müssen daher in korrekter Syntax des ENTER-PROC-Kommandos angegeben werden.

Beispiel:

```
.UTM START ENTER-PROC-INPUT='START-DYN-PROC,(PAR1=<par1>,PAR2=<par2>'
.UTM START ENTER-PROC-INPUT=',PAR3=<par3>),LOGGING=*NO'
```

Damit wird folgendes Kommando abgesetzt:

```
ENTER-PROC START-DYN-PROC,(PAR1=<par1>, PAR2=<par2>,
PAR3=<par3>),LOGGING=*NO
```

Bei Syntax-Fehlern in diesem Kommando wird die Meldung K048 ausgegeben und keine Folgetask gestartet.
Die gleichzeitige Angabe der Parameter ENTER-PROC-INPUT und STARTNAME wird mit der Meldung K039 abgewiesen, der Start der Anwendung wird abgebrochen.

> **i** Die gleichzeitige Angabe der Parameter ENTER-PROC-INPUT und STARTNAME wird mit der Meldung K039 abgewiesen. Der Start der Anwendung wird abgebrochen.

OTRACE =

Ein-/Ausschalten der Tracefunktion von OSS.
Der OSS-Trace wird zur Diagnose bei Problemen mit OSI TP-Verbindungen der Anwendung benötigt. Sehen Sie dazu auch das openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“ und das Handbuch zu OSS.

ON

Schaltet die OSS-Tracefunktion beim Start der Anwendung ein.

Es werden die Trace-Records der Typen SPI, INT, OSS, SERV und PROT protokolliert. Beim Einschalten der OSS-Tracefunktion erzeugt jede Task der Anwendung seine eigene Trace-Datei.

(SPI,INT,OSS, SERV,PROT)

Schaltet die OSS-Tracefunktion beim Start der Anwendung ein. Es werden die Trace-Records des angegebenen Typs protokolliert. Die Reihenfolge, in der die Trace-Records angegeben werden, ist beliebig.

SPI
: Das XAP-TP System Programming Interface wird protokolliert.

INT
: Der interne Ablauf im XAP-TP-Baustein wird protokolliert.

OSS
: Die OSS-Aufrufe werden protokolliert.

SERV
: Die OSS-internen Trace Records vom Typ O_TR_SERV werden protokolliert.

PROT
: Die OSS-internen Trace Records vom Typ O_TR_PROT werden protokolliert.

OFF

Die OSS-Tracefunktion bleibt beim Start der Anwendung ausgeschaltet.

Standard: OFF

PASSWORD = connection-password

ein 4 Zeichen langes Kennwort, das der Benutzer zum Aufbau der logischen Terminal-Verbindung mit der Anwendung angeben muss (siehe Abschnitt „Standard-Anmeldeverfahren für Terminals“).

Standard: kein Kennwort (X’00000000’)

ROOTNAME = rootname

Der Parameter ist Pflicht, wenn der ROOT-Tabellenmodul nachgeladen werden soll. *rootname* ist der PLAM-Elementname des ROOT-Tabellenmoduls in der bei TABLIB= angegebenen Bibliothek.

STARTNAME =

enterfile

Angabe der ENTER-Datei. Wird dieser Parameter angegeben, so werden die Folgetasks einer UTM-Anwendung mittels eines ENTER-Kommandos gestartet.

Die ENTER-Datei muss unter der Benutzerkennung katalogisiert sein, unter der die Anwendung gestartet wurde. *enterfile* darf maximal 54 Byte lang sein.

enteroperand

Operand des BS2000-Kommandos ENTER. openUTM startet die Folgetasks und setzt dabei die angegebenen Operanden ein. Schreibweise und Wirkung der angegebenen Operanden entsprechen den Operanden des ENTER-Kommandos. Die Operanden müssen im ISP-Format eingegeben werden. Die Zeichenfolge zwischen den Hochkommata (*'enterfile, enteroperand,...'*) darf maximal 200 Zeichen lang sein.

> **i** Die gleichzeitige Angabe der Parameter ENTER-PROC-INPUT und STARTNAME wird mit der Meldung K039 abgewiesen. Der Start der Anwendung wird abgebrochen.

START-SSL-PROC = enter-ssl-proc-input

Dieser Parameter wird nur benötigt wenn man einen BCAMAPPL mit SECURE generiert hat(s. in Kapitel "UTM-Anwendung starten Starten der Anwendung mit TLS-Verbindungen)

Angabe der eigenen START-Prozedur für den SSL-Proxy und ggfs. Parameter für den ENTER-PROCEDURE-Aufruf, ohne die zum Start des SSL-Proxy notwendigen Prozedur-Parameter (diese werden von UTM automatisch erzeugt und beim Aufruf der Prozedur mitgegeben).

Die Länge von enter-ssl-proc-input darf maximal 255 Zeichen betragen. Dieser Parameter kann mehrfach angegeben werden, die Angabe muss jeweils inHochkommata erfolgen. Die in mehreren START-SSL-PROC-Anweisungen angegebenParameter werden in der Reihenfolge ihrer Angabe zusammengesetzt.

Die maximale Länge der aufgesammelten Angaben beträgt 2000 Zeichen.

Sie sollten den Job für SSL-Proxy wie die UTM-Tasks ohne CPU-Zeitbeschränkung starten. Aus diesem Grund sollten Sie CPU-LIMIT=NO(bzw. TIME=NTL) setzen bzw. die Jobklasse entsprechend definieren.

Wird die CPU-Zeit beschränkt, d.h. CPU-LIMIT ungleich NO gesetzt, und tritt für den Job ein CPU-Time-Runout auf, dann wird der SSL-Proxy abnormal beendet.

Bitte beachten Sie, dass für den Ablauf der Enter-Tasks ohne CPU-Zeitbegrenzung eventuell ein NTL-Berechtigung(**N**o**T**ime**L**imit) im Benutzereintrag für die betreffende Abrechnungsnummer notwendig ist.

Beispiel:

.UTM START START-SSL-PROC='OWN.START.SSL,JOB-NAME=ownnam,JOB-CLASS=jcbname'

Damit wird folgendes Kommando abgesetzt:

ENTER-PROC OWN.START.SSL,JOB-NAME=ownnam,JOB-CLASS=jcbnam, PROC-PAR=(par1,par2,par3..5)

Die Prozedur-Parameter (PROC-PAR=(...) werden von UTM hinzugefügt.

Bei Syntax-Fehlern in diesem Kommando kommt eine K099-Meldung, der Start der Anwendung wird fortgesetzt (aber kein SSL-Proxy gestartet).

STXIT =

STXIT-Routinen ein-/ausschalten.

ON — Beim Start der Anwendung wird die UTM-STXIT eingeschaltet.

Standard: ON

OFF — Die UTM-STXIT bleibt ausgeschaltet. Dies ist nur möglich für UTM-Anwendungen, die im Dialog gestartet werden.

STXIT-LOG =

Einschalten des erweiterten STXIT-Loggings bei Problemen mit der STXIT-Behandlung. Es werden mehrere K099-Meldungen auf SYSOUT ausgegeben.

ON — Beim Start der Anwendung wird das STXIT-Logging eingeschaltet.

OFF — Beim Start der Anwendung bleibt das STXIT-Logging ausgeschaltet.

Standard: OFF

SYSPROT =

(interval, filename-prefix)

Systemdateien SYSOUT und SYSLST umschalten

interval — Umschaltintervall in Tagen
Standard: 0 (kein Intervall, das Umschalten erfolgt nicht zeitgesteuert, sondern nur auf Anforderung)

Maximalwert: 364

filename-prefix — Präfix für den neuen Dateinamen der umgeschalteten System-Dateien.*filename-prefix* darf maximal 17 Zeichen lang sein, siehe auch "Systemdateien SYSOUT und SYSLST".

Standard : Name der Anwendung, der bei der KDCDEF-Generierung in MAX APPLINAME festgelegt wurde.

Eine vollständige Beschreibung zur Umschaltung der System-Protokolldateien finden Sie im Abschnitt „Systemdateien SYSOUT und SYSLST“.

TABLIB = libname

Der Parameter ist Pflicht, wenn der ROOT-Tabellenmodul nachgeladen werden soll. Die Bibliothek *libname* muss dann das Objekt des ROOT-Tabellenmoduls enthalten.

TASKS = number

Anzahl der BS2000-Tasks, die für die Anwendung gestartet werden sollen.

Standard: in MAX...,TASKS=*number* festgelegte Anzahl

Minimalwert: 1 *)
Maximalwert: in MAX...,TASKS=*number* festgelegte Anzahl

*) Falls die Anwendung mit Program Wait generiert ist (d.h. wenn entweder eine TAC-Klasse oder ein TAC mit PGWT=YES generiert ist) dann muss für den Startparameter TASKS mindestens 2 angegeben werden.

> ¡ Zusätzlich zu der bei TASKS festgelegten Anzahl von Tasks werden von UTM für eine Anwendung weitere Tasks gestartet, die als UTM-System-Task bezeichnet werden. Die UTM-System-Prozesse sollen Anwendungen, die unter Last laufen, reaktionsfähig halten. Siehe auch Manual "Konzepte und Funktionen" , Kapitel Prozesskonzept.

Die System-Prozesse bearbeiten nur ausgewählte Aufträge, die in erster Linie durch kurze Laufzeiten gekennzeichnet sind. Beim Start einer Anwendung werden von UTM - abhängig von der Anzahl gestarteter Tasks (TASKS= number) - bis zu drei zusätzliche UTM-System-Prozesse für die Anwendung gestartet.

TASKS-IN-PGWT = number

Maximale Anzahl der Tasks, in denen gleichzeitig Teilprogramme mit blockierenden Aufrufen wie z.B. der KDCS-Aufruf PGWT ablaufen dürfen (Operand PGWT= in den KDCDEF-Anweisungen TAC und TACCLASS).

Standard: in MAX...,TASKS-IN-PGWT=*number* festgelegte Anzahl.

Minimalwert: 1 falls MAX...,TASKS-IN-PGWT > 0 generiert wurde, sonst 0.

Maximalwert: in MAX...,TASKS-IN-PGWT=*number* festgelegte Anzahl.

TESTMODE =

Testmodus ein-/ausschalten. Sehen Sie hierzu auch openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“, Fehlerdiagnose.

ON — Beim Start der Anwendung soll der Testmodus eingeschaltet werden.

Im Testmodus werden zusätzliche UTM-interne Plausibilitätsprüfungen durchgeführt und Trace-Informationen im internen Trace-Bereich aufgezeichnet.

Sie müssen den Testmodus einschalten, wenn Sie zu bestimmten Ereignissen UTM-Dumps schreiben wollen, ansonsten sollten Sie den Testmodus nur auf Empfehlung des System Centers einschalten.

OFF — Beim Start der Anwendung soll der Testmodus ausgeschaltet bleiben. In bestimmten Fällen werden bei TESTMODE=OFF Dumps nach PEND ER unterdrückt.

Standard: OFF

FILE — Beim Start der Anwendung wird der Testmodus eingeschaltet. Zusätzlich werden bei jedem Überlauf des internen Trace-Bereichs die Diagnosedaten auf Datei geschrieben, um einen evtl. Diagnosedatenverlust zu vermeiden.
Der Dateiname setzt sich zusammen aus dem Basisnamen *filebase* und der TSN der jeweiligen Task, d.h. es wird bei einer UTM-Produktiv-Anwendung pro Task folgende Datei angelegt:

filebase.KTATRC.*tsn*

TX-TRACE =

Ein-/Ausschalten der TX-Tracefunktion (= Tracefunktion der X/Open-Schnittstelle TX), siehe auch openUTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

Zum Namen der Trace-Dateien siehe „Trace-Dateien“.

ERROR — Beim Start der Anwendung wird die TX-Tracefunktion mit Level ERROR eingeschaltet. Es werden nur Fehler protokolliert.

INTERFACE — Beim Start der Anwendung wird die TX-Tracefunktion mit Level INTERFACE eingeschaltet. Der Level INTERFACE umfasst den Level ERROR, zusätzlich werden TX-Aufrufe protokolliert.

FULL — Beim Start der Anwendung wird die TX-Tracefunktion mit Level FULL eingeschaltet. Der Level FULL umfasst den Level INTERFACE, zusätzlich werden alle KDCS-Aufrufe, auf die die TX-Aufrufe abgebildet werden, protokolliert.

DEBUG — Beim Start der Anwendung wird die TX-Tracefunktion mit Level DEBUG eingeschaltet. Der Level DEBUG umfasst den Level FULL, zusätzlich werden Diagnose-Informationen protokolliert.

OFF — Beim Start der Anwendung bleibt die Tracefunktion der TX-Schnittstelle ausgeschaltet. Standard: OFF

UTM-MSG-DATE =

gibt an, ob UTM-Meldungen mit oder ohne Datum/Uhrzeit auf SYSOUT/SYSLST ausgegeben werden.

YES — Alle UTM-Meldungen auf SYSOUT/SYSLST werden um ein Datum/Uhrzeit-Präfix ergänzt.

Standard: YES

NO — Die UTM-Meldungen erhalten kein Datum/Uhrzeit-Präfix.

XATMI-TRACE =

Ein-/Ausschalten der XATMI-Tracefunktion (= Tracefunktion der X/Open-Schnittstelle XATMI), siehe auch openUTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

Zum Namen der Trace-Dateien siehe „Trace-Dateien“.

ERROR — Beim Start der Anwendung wird die XATMI-Tracefunktion mit Level ERROR eingeschaltet. Es werden nur Fehler protokolliert.

INTERFACE — Beim Start der Anwendung wird die XATMI-Tracefunktion mit Level INTERFACE eingeschaltet. Der Level INTERFACE umfasst den Level ERROR, zusätzlich werden alle XATMI-Aufrufe protokolliert.

FULL — Beim Start der Anwendung wird die XATMI-Tracefunktion mit Level FULL eingeschaltet. Der Level FULL umfasst den Level INTERFACE, zusätzlich werden alle KDCS-Aufrufe, auf die die XATMI-Aufrufe abgebildet werden, protokolliert.

DEBUG — Beim Start der Anwendung wird die XATMI-Tracefunktion mit Level DEBUG eingeschaltet.

Der Level DEBUG umfasst den Level FULL, zusätzlich werden Diagnose-Informationen protokolliert.

OFF — Beim Start der Anwendung bleibt die Tracefunktion der XATMI-Schnittstelle ausgeschaltet.

Standard: OFF

Trace-Dateien

Die Trace-Sätze der ADMI-, CPI-C-, TX, und XATMI-Tracefunktion werden standardmäßig in eine der folgenden Dateien geschrieben:

KDC.TRC.trace-type.appliname.tsn

trace-type

Kennzeichnet den Trace-Typ:

ADMI ADMI-Trace (Linkname: KDCADMI)

CPIC CPI-C-Trace (Linkname: KDCCPIC)

TX TX-Trace (Linkname: KDCTX)

XATMI XATMI-Trace (Linkname: KDCXATMI)

appliname

Name der Anwendung

tsn

TSN der UTM-Task (4-stellig)

Über Linknamen können Sie auch eigene Trace-Dateien zuweisen.

5.1.2 Startparameter für das Datenbanksystem

Die Schreibweise und die Bedeutung dieser Parameter ist in den Benutzerhandbüchern der entsprechenden DB-Systeme beschrieben. Zusätzliche Informationen finden Sie im Abschnitt „UTM-Datenbank-Anwendung starten und beenden“.

5.1.3 Startparameter für das Formatierungssystem

Die Schreibweise und die Bedeutung dieser Startparameter sind im Benutzerhandbuch des Formatierungssystems (FHS) beschrieben. openUTM akzeptiert Startparameter für das Formatierungssystem nur dann, wenn dieses in der KDCDEF-Anweisung FORMSYS bei der UTM-Generierung angegeben wurde.

5.2 Starten der Anwendung

Das Anwendungsprogramm wird durch das Kommando START-EXECUTABLE-PROGRAM gestartet. Dabei gilt Folgendes:

- Falls shareable Teile der Anwendung oder der von der Anwendung benötigten Laufzeitsysteme in den Systemspeicher geladen werden sollen, so muss dies noch vor dem Start des Anwendungsprogramms durch den Administrator veranlasst werden.
- Das START-EXECUTABLE-PROGRAM-Kommando bewirkt, dass der statisch gebundene Teil des Anwendungsprogramms in den Arbeitsspeicher geladen wird. Alle Lademodule des Anwendungsprogramms, die mit LOAD-MODE=STARTUP oder LOAD-MODE=(POOL,...) generiert wurden, werden beim Start der Anwendung als eigenständige Einheiten nachgeladen.
- Können einzelne Lademodule nicht geladen werden, so wird i.a. der Start der Anwendung fortgesetzt. Soll im Betrieb der Anwendung ein Programm angesprungen werden, das nicht geladen werden konnte, so führt dies zum Aufruf des Event-Services BADTACS oder zu einem PEND ER. Wenn die Event-Services MSGTAC, SIGNON oder die Event-Exits START, SHUT, INPUT und FORMAT oder das Administrations-Teilprogramm oder AREAs nicht geladen werden können, wird der Start der Anwendung mit einer Fehlermeldung abgebrochen.
- Beim Starten einer zusätzlichen Task für die Anwendung sowie nach einem PEND ER prüft openUTM anhand der Generierungsinformationen, ob das in dieser Task geladene Anwendungsprogramm mit dem in den zuvor gestarteten Tasks übereinstimmt. Bei fehlender Übereinstimmung bricht openUTM den Start der Folge-Task mit einer Fehlermeldung ab.

Startkommandos für das Anwendungsprogramm

Das Anwendungsprogramm wird gestartet mit:

```
/START-EXECUTABLE-PROGRAM FROM-FILE=*LIB-ELEM  -
/               (  LIB=llm-plamlib             -
/                , ELEM=start-llm              -
/               )
```

Das Anwendungsprogramm *start-llm* muss als Element vom Typ L in einer Programmbibliothek *llm-plamlib* bereitgestellt werden.

Wenn Sie ein Lademodul mit ALTERNATE-LIBRARIES=YES generiert haben, dann müssen Sie vor dem Start des Anwendungsprogramms der Bibliothek des betreffenden Laufzeitsystems einen Linknamen (BLSLIB*nn* mit 00<=*nn* <=99) zuweisen.

Sie können sich mit dem BS2000-Kommando

```
/MODIFY-DBL-DEFAULT PRIORITY=*FORCED, SCOPE=*ALL(PROGRAM-MAP=...)
```

eine DBL-Liste über die dynamischen Ladevorgänge während des Betriebs der Anwendung ausgeben lassen. Da dies aber den Startvorgang bzw. den Programmaustausch verzögert, sollten Sie von dieser Möglichkeit nur im Test- oder Fehlerfall Gebrauch machen.

Wenn Teile des Anwendungsprogramms zu einem späteren Zeitpunkt nachgeladen werden sollen, müssen Sie beim Starten die folgenden Operanden angeben:

```
/     ,DBL-PARAMETERS = *PARAMETERS(                                    -
/          ,LOADING    = *PARAMETERS(                                   -
/                          PROGRAM-MODE = *ANY                           -
/                         ,LOAD-INFORMATION = *REFERENCES)               -
/          ,RESOLUTION = *PARAMETERS(                                    -
/                          ALTERNATE-LIBRARIES = *BLSLIB##               -
/                         ,AUTOLINK = *ALTERNATE-LIBRARIES )             -
/          ,ERROR-PROCESSING=*PARAMETERS(                                -
/                          UNRESOLVED-EXTRNS=*DELAY                      -
/                         ,ERROR-EXIT = *NONE))
```

Die folgende Liste zeigt die Möglichkeiten, wie Sie die Autolink-Funktion beeinflussen können:

- Wenn Sie die Autolink-Funktion während des Starts nicht benötigen und Sie im Start-LLM unbefriedigte Externverweise haben, sollten Sie die Funktion mit AUTOLINK=*NO unterdrücken.
- Mit AUTOLINK=*YES,ALTERNATE-LIBRARIES=*NO wird die Bibliothek aus dem Ladeaufruf durchsucht.
- Mit AUTOLINK=*ALTERNATE-LIBRARIES,ALTERNATE-LIBRARIES=*TASKLIB/*BLSLIB## werden Tasklib und /oder die BLS-Libs durchsucht.
- Mit AUTOLINK=*YES,ALTERNATE-LIBRARIES=*TASKLIB/*BLSLIB## werden die Bibliothek aus dem Ladeaufruf sowie Tasklib und/oder BLS-Libs durchsucht.

Wenn das Start-LLM beim Start keine offenen Externverweise auf den Shared Code besitzt, sollte das Suchen des DBL im Shared Code mit dem Startoperanden SHARE-SCOPE=*NONE unterbunden werden.

Beim Start der Anwendung lädt openUTM zunächst alle Lademodule, die shareable in Common Memory Pools (LOAD-MODE=POOL) gehalten werden sollen. Dabei werden zuerst die Common Memory Pools geladen, die mit SCOPE=GLOBAL generiert wurden und danach diejenigen mit SCOPE=GROUP, jeweils in der bei der Generierung angegebenen Reihenfolge. Anschließend lädt openUTM alle Lademodule, die mit LOAD-MODE=STARTUP generiert wurden, ebenfalls in der Reihenfolge, in der Sie die LOAD-MODULE-Anweisungen bei der UTM-Generierung angegeben haben.

Lademodule, die mit LOAD-MODE=ONCALL generiert wurden, werden erst beim Aufruf eines Teilprogramms dieses Lademoduls geladen.

> **i** Verschiedene UTM-Anwendungen sollten möglichst unter unterschiedlichen BS2000-Benutzerkennungen gestartet werden, um Fehler zu vermeiden, die wegen gleicher Modulnamen in shareable Teilen auftreten können. Module, die von mehreren Anwendungen benutzt werden, sollten daher in anwendungsglobale Common Memory Pools oder in nichtprivilegierte Subsysteme geladen werden.
>
> Mit openUTM wird auch eine Beispiel-Startprozedur ausgeliefert, siehe Abschnitt „Beispielprozeduren“.

Starten einer Anwendung mit TLS-Verbindungen

Ist für eine UTM-Anwendung ein BCAMAPPL mit T-PROT=(SOCKET,...,SECURE) generiert, dann erfolgt die Kommunikation über diesen Transportsystemzugangspunkt über Transport Layer Security (TLS). Dazu wird von openUTM beim Start einer so generierten Anwendung eine zusätzliche Task gestartet, in der ein Reverse Proxy zum Ablauf kommt, über den in Form eines TLS Termination Proxy alle TLS-Verbindungen abgewickelt werden. Die Task mit dem Reverse Proxy wird mit dem Jobnamen "SSLPROXY" gestartet. Beim Beenden der Anwendung wird auch der Reverse Proxy beendet.

Der Reverse Proxy schreibt eine SYSOUT-Datei mit folgendem Muster: <APPLNAME>.<JOBNAME>.<TSN>. YYYY-MM-DD-HH:MM:SS . Diese Datei wird um Mitternacht umgeschaltet.

Der Reverse Proxy kommuniziert mit HTTPS-Clients über TLS-Verbindungen und mit der UTM-Anwendung über eine Rechner-lokale Socket-Verbindung. Der Reverse Proxy muss dazu auf dem gleichen Rechner ablaufen wie die UTM-Anwendung.

Als Voraussetzung für den Start eines Reverse Proxy für eine UTM-Anwendung muss eine Jobvariable mit dem Basisnamen der KDCFILE katalogisiert sein. Dazu wird das Produkt JV ("Jobvariable") benötigt. Diese Jobvariable dient zum geordneten Beenden des Proxy Prozesses.

openUTM startet die Task mit dem Reverse Proxy über eine ENTER-Prozedur, die mit dem Namen START-SSL-PROXY in der Bibliothek SYSLIB.UTM.070.SSL abgelegt ist. Bei Bedarf kann diese Prozedurdatei an die Bedürfnisse des Anwenders angepasst werden.

In diesem Falle sollte die Prozedur aus der Bibliothek geholt und entsprechend angepasst werden.

Beim Starten der Anwendung wird der Startparameter „START-SSL-PROC“ mit dem Namen dieser Prozedur und evtl. gewünschten Ablauf-Parametern angegeben. Die benötigten Prozedur-Parameter wie filebasename, Listenerports etc. werden von openUTM versorgt.

Ist kein Startparameter angegeben so wird die Prozedur aus der o.g. Bibliothek gestartet.

Die Task mit dem Reverse Proxy liest SSL-Konfigurationsparameter von der SAM-Datei

<filebase>.SSL.CONF

Eine solche Datei mit Namen FILEBASE.SSL.CONF wird von openUTM als Beispiel mit ausgeliefert. Auch diese befindet sich in der Bibliothek SYSLIB.UTM.070.SSL. Sie muss vom Anwender für seine Anwendung umbenannt und in die Kennung kopiert werden, in der die UTM-Anwendung gestartet wird. Vor dem Start der Anwendung muss der Anwender die Einträge in dieser Datei an seine Umgebung und Bedürfnisse anpassen.

Folgende Parameter können in dieser Datei konfiguriert werden.

CACertificateFile

In der Option *CACertificateFile* wird eine Datei angegeben, die die für die Authentifizierung des Reverse Proxy erforderlichen CA-Zertifikate im PEM-Format enthält.

CipherSuite

Mit der Option *CipherSuite* wird eine Verschlüsselungsverfahren-Vorzugsliste spezifiziert.

Falls diese Option nicht angegeben wird, dann wird eine voreingestellte Vorzugsliste verwendet.

DSACertificateFile

In der Option *DSACertificateFile* wird eine Datei angegeben, die das DSA-basierte X.509-Server-Zertifikat im PEM-Format enthält.

Diese Datei kann auch den privaten DSAServer-Schlüssel enthalten. In der Regel werden aber Zertifikat und Schlüssel in getrennten Dateien abgelegt. In diesem Fall wird die Schlüsseldatei mithilfe der Option *DSAKey-File* spezifiziert.

DSAKeyFile

In der Option *DSAKeyFile* wird eine Datei angegeben, die den privaten DSA-Server-Schlüssel im PEM-Format enthält.

Wenn sowohl X.509-Server-Zertifikat als auch privater Server-Schlüssel innerhalb derselben Datei enthalten sind, dann braucht die Option D*SAKeyFile* nicht angegeben werden.

RSACertificateFile

In der Option *RSACertificateFile* wird eine Datei angegeben, die das RSA-basierte X.509-Server-Zertifikat im PEM-Format enthält.

Diese Datei kann auch den privaten RSA-Server-Schlüssel enthalten. In der Regel werden aber Zertifikat und Schlüssel in getrennten Dateien abgelegt. In diesem Fall wird die Schlüsseldatei mithilfe der Option *RSAKeyFile* spezifiziert.

RSAKeyFile

In der Option *RSAKeyFile* wird eine Datei angegeben, die den privaten RSA-Server-Schlüssel im PEM-Format enthält.

Wenn sowohl X.509-Server-Zertifikat als auch privater Server-Schlüssel innerhalb derselben Datei enthalten sind, dann braucht die Option *RSAKeyFile* nicht angegeben werden.

ECDSACertificateFile

In der Option *ECDSACertificateFile* wird eine Datei angegeben, die das ECDSA-basierte X.509-Server-Zertifikat im PEM-Format enthält.

ECDSAKeyFile

In der Option *ECDSAKeyFile* wird eine Datei angegeben, die den privaten ECDSA-Server-Schlüssel im PEM-Format enthält.

Weitere Einzelheiten zur Syntax und Bedeutung der Optionen, sowie zum Erzeugen der Zertifikat- und Schlüsseldateien sind im Handbuch "**interNet Services Benutzer V3.4B**" im Kapitel "SSL",

sowie im Handbuch "**interNet Services Administrator V3.4B**" im Kapitel "TELNET Konfiguration" beschrieben.

5.3 Kaltstart und Warmstart

Man versteht bei openUTM unter:

- Kaltstart:
  Starten nach normaler Beendigung der UTM-Anwendung oder nach Neugenerierung.
- Warmstart:
  Starten nach abnormaler Beendigung der UTM-Anwendung.

Kaltstart bei openUTM

Vor dem ersten Start einer Anwendung haben Sie die KDCFILE mit dem Generierungstool KDCDEF erzeugt. Nach einer Neu-Generierung der KDCFILE oder wenn eine UTM-Anwendung zuvor normal beendet wurde, führt openUTM beim nächsten Start der Anwendung einen Kaltstart durch. Nach erfolgreichem Start meldet openUTM:

```
K051 Kaltstart für Anwendung <anwendungsname> mit UTM V07.0A00 / betriebssystem-typ
erfolgreich
```

Warmstart bei openUTM

Wenn eine UTM-Anwendung abnormal beendet wurde, führt openUTM beim nächsten Start dieser Anwendung einen Warmstart durch. Beim Warmstart bringt openUTM die KDCFILE in einen konsistenten Zustand. Nach erfolgreichem Start meldet openUTM:

```
K050 Warmstart für Anwendung <anwendungsname> mit UTM V07.0A00 / betriebssystem-typ
erfolgreich.
```

Dabei ist zu beachten, dass sich UTM-S und UTM-F im Umfang der Wiederanlauf-Funktionen unterscheiden. Sehen Sie dazu auch im openUTM-Handbuch „Konzepte und Funktionen“ nach.

5.4 Fehlermeldungen beim Start

Wird der Start einer UTM-Anwendung oder einer Task wegen eines Fehlers abgebrochen, gibt openUTM in der Regel die Meldungen K049 und/oder K078 aus. Die Meldung K078 kann in mehreren Varianten auftreten. Die Bedeutung dieser Meldungen und der darin enthaltenen Returncodes sind ausführlich im openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“ erklärt.

Es wird zum nächsten Kommando /SET-JOB-STEP oder zum /EXIT-JOB- bzw. /LOGOFF-Kommando bzw. bei der Verwendung einer SDF-Prozedur bis zum nächsten IF-BLOCK-ERROR Kommando in der Prozedur verzweigt.

Startfehler können beim Start einer jeden Task auftreten.

5.5 Nach abnormalem Anwendungsende erneut starten

openUTM bietet die Möglichkeit, eine Anwendung nach einem abnormalen Anwendungsende automatisch neu starten zu lassen.

> ¡ Als Voraussetzung für das im Folgenden geschilderte Vorgehen wird das Produkt JV („Jobvariablen“) benötigt.

Wird vor dem Start der Anwendung eine Jobvariable katalogisiert, die als Namen den Basisnamen der KDCFILE (inklusive Catid und Userid) trägt, so wird diese Jobvariable von openUTM benutzt.

Die erste Position der Jobvariablen kann die Werte R oder T annehmen:

R: UTM-Anwendung läuft

T: UTM-Anwendung ist beendet

Die zweite Position der Jobvariablen kann die Werte N oder A annehmen:

N: normal beendet

A: abnormal beendet

Diese Jobvariable kann dazu benutzt werden, den Ablauf einer UTM-Anwendung durch einen ENTER-Job zu überwachen, der im Fall, dass sich die Anwendung abnormal beendet, die Anwendung neu startet. Wenn die erste Spalte der Jobvariablen auf ’T’ steht, wird abhängig vom Wert der zweiten Jobvariablenspalte entweder dieser ENTER-Job beendet (bei ’N’) oder die UTM-Anwendung neu gestartet (bei ’A’).

Der ENTER-Job kann folgenden Aufbau haben:

```
/SET-LOGON-PARAMETERS
/.WAIT1 REMARK
/WAIT-EVENT UNTIL=*JV(CONDITION=((jobvariable-name,1,1)='T'), -
/           TIME-LIMIT=32767,TIMEOUT-LABEL=WAIT1)
/SKIP-COMMANDS TO-LABEL=END,IF=*JV(CONDITION=((jobvariable-name,2,1)='N'))
/ENTER-JOB enterfile,CPU-LIMIT=ttt,JOB-CLASS=job-class
/WAIT-EVENT UNTIL=*JV(CONDITION=((jobvariable-name,1,1)='R'))
/WAIT-EVENT UNTIL=*JV(CONDITION=*NONE,TIME-LIMIT=600,TIMEOUT-LABEL=WAIT2)
/.WAIT2 REMARK
/SKIP-COMMANDS TO-LABEL=END,IF=*JV(CONDITION=((jobvariable-name,1,2)='TA'))
/SKIP-COMMANDS TO-LABEL=WAIT1
/.END REMARK
/EXIT-JOB
```

Beschreibung des ENTER-Jobs

- Das erste WAIT-EVENT-Kommando lässt den Job warten, bis sich die UTM-Anwendung beendet hat (‘T‘ im ersten Feld der JV). Bei einer abnormalen Beendigung (‘A‘ im zweiten Feld der JV) wird der ENTER-Job zum erneuten Starten der UTM-Anwendung angestoßen, bei normaler Beendigung wird auch der Überwachungsjob beendet.
- Mit dem zweiten WAIT-EVENT-Kommando wird gewartet, bis die UTM-Anweisung wieder läuft.
- Mit dem dritten WAIT-EVENT-Kommando wird dann zehn Minuten (600 sek) gewartet, bis abgefragt wird, ob sich die UTM-Anwendung erneut abnormal beendet hat. Wenn ja, wird der Überwachungsjob beendet, und die UTM-Anwendung wird nicht hochgefahren. Wenn nicht, wird zum ersten WAIT-EVENT zurückverzweigt, und der Überwachungsjob legt sich erneut auf die Lauer. Damit wird verhindert, dass die UTM-Anwendung bei einem permanenten Fehler ständig beendet und wieder hochgefahren wird.

5.6 Grundstruktur für eine SDF-Startprozedur

Das folgende Beispiel zeigt eine SDF-Startprozedur. Wie für die Nutzung von SDF-P empfohlen, werden BEGIN-BLOCK, END-BLOCK sowie IF-BLOCK-ERROR verwendet.

```
/SET-PROCEDURE-OPTIONS                                -
/        CALLER               = ANY                   -
/       ,IMPLICIT-DECLARATION = YES                   -
/       ,LOGGING              = YES                   -
/       ,INTERRUPT-ALLOWED    = YES                   -
/       ,INPUT-FORMAT         = FREE                  -
/       ,DATA-ESC             = STD                   -
/       ,SYS-FILE             = STD                   -
/       ,DATA-ERROR           = YES                   -
/       ,JV-REPLACEMENT       = AFTER-BUILTIN-FUNCTION
/BEGIN-PARAMETER-DECLARATION
/&* ------------------------------------------------
/&* here you can declare your Procedure-Parameters
/&* ------------------------------------------------
/END-PARAMETER-DECLARATION
/&* -----------------------------------
/&* here is place for your variables
/&* -----------------------------------
/ SET-FILE-LINK LINK-NAME=SYSLOG,FILE-NAME=<applname>.SYSLOG
/ MODIFY-TEST-OPTION DUMP=YES
/ ASSIGN-SYSDTA TO-FILE=*SYSCMD
/ ASSIGN-SYSOUT TO-FILE=SYSOUT.&(TSN()).<applname>
/&* ---------------------------------------------------------------
/&*  Here follows the section with the SET-FILE-LINK commands if
/&*   at least one LOAD-MODULE is defined with
/&*   ALTERNATE-LIBRARIES = YES.
/&*  If you need RFA connections, or an AID connection (AID-FE)
/&*   for testing an UTM production application or other
/&*   connections please insert here the commands.
/&* ---------------------------------------------------------------
/REPEAT: BEGIN-BLOCK
/ SHOW-FILE-LINK LINK-NAME=KDCDUMP
/&* ---------------------------------------------------------------
/&*  If you want to do some actions after PEND ER, then you must
/&*   insert commands here. (E.g. prepare an UTM dump.)
/&* ---------------------------------------------------------------
/ REMOVE-FILE-LINK LINK-NAME=KDCDUMP
/ GOTO EXEC
/ IF-BLOCK-ERR; END-IF
/&* ------------------------------------------------
/&* here is place for actions after KDCAPPL PROG=NEW
/&* ------------------------------------------------
```

```
/EXEC:
/ MOD-DBL-DEFAULTS SCOPE=*CMD-CALLS(               -
/      LOAD=*PAR(LOAD-INF=*REF),                   -
/      RESOLUTION=*PAR(SHARE-SCOPE=*NONE),         -
/      ERROR-PROC=*PAR(UNRESOLVED-EXTRNS=*DELAY),  -
/      REPORT=*PAR(MES-CONTR=*WARN,                -
/                  PROG-MAP=*SYSLST) )
/ START-EXEC-PROG FROM-FILE=*LIBRARY-ELEMENT (     -
/                    LIBRARY  = <libname>          -
/                    , ELEMENT  = <elem>)
.UTM START FILEBASE  = <applname>
.UTM START ROOTNAME  = <rootname>
.UTM START TABLIB    = <root-lib>
/&* ------------------------------------------------
/&* in this section you can insert the parameter(s)
/&* for node-recovery  for example:
/&* .UTM START NODE-TO-RECOVER=<nodename>,RESET-PTC=Y/N
/&* ------------------------------------------------
.UTM START ENTER-PROC-INPUT='STRT-ENTER-PROC'
/&* ------------------------------------------------
/&* if your enter-procedure has parameters you have to
/&* change the last line with correct syntax
/&* if necessary duplicate the parameter-line
/&* ------------------------------------------------
.UTM START TASKS     = 3
.UTM START TASKS-IN-PGWT     = 0
.UTM START ASYNTASKS = 2
.UTM START TESTMODE  = OFF
.UTM START STXIT     = ON
.UTM START BTRACE    = OFF
.UTM START OTRACE    = OFF
.UTM START DB-CONNECT-TIME = 0
.UTM END
/&* ------------------------------------------------------------
/&*  Here follows the section with the data base and format system
/&*   parameters (if necessary).
/&* ------------------------------------------------------------
.FHS MAPLIB          = <maplib>
END
/ GOTO REPEAT
/ IF-BLOCK-ERR
/ SHOW-FILE-LINK LINK-NAME=KDCTRMAP
/&* ----------------------------------------------------------
/&*  If you want to do some actions after abnormal application
/&*   termination, then you can put some commands here.
/&* ----------------------------------------------------------
/ END-IF
/ END-BLOCK REPEAT
/ GOTO EXIT
/ IF-BLOCK-ERR; END-IF
/&* ----------------------------------------------------
/&*  Here you can insert commands to execute after normal
/&*   application termination.
/&* ----------------------------------------------------
/EXIT:
/ REMARK
/EXIT-PROC
```

6 UTM-Anwendung beenden

Eine UTM-Anwendung kann

- normal beendet werden durch Administration
- abnormal beendet werden aufgrund von Betriebsmittelengpässen, internen Fehlern in openUTM oder durch Administration

openUTM beendet alle Prozesse mit TERMJ, d.h. es wird zum nächsten /SET-JOB-STEP-Kommando oder zum /EXIT-JOB- bzw. /LOGOFF-Kommando bzw. bei der Verwendung einer SDF-Prozedur bis zum nächsten IF-BLOCK-ERROR Kommando in der Startprozedur verzweigt.

In der Startprozedur können Sie abfragen, ob die UTM-Anwendung normal oder nicht normal beendet wurde. Für die Abfrage gibt es folgende Möglichkeiten:

- Auftragsschalter 3
- TFT-Eintrag (LINK-NAME=KDCTRMAP)
- Jobvariable

Bei Nutzung von SDF-P sollte mit BEGIN-/END-BLOCK und IF-BLOCK-ERROR gearbeitet werden, siehe Abschnitt „Grundstruktur für eine SDF-Startprozedur“.

6.1 UTM-Anwendung per Administration normal beenden

Eine UTM-Anwendung beendet der UTM-Administrator normal, indem er z.B. folgendes UTM-Administrationskommando an einem Terminal eingibt:

KDCSHUT GRACE,TIME=time

oder

KDCSHUT WARN,TIME=time

oder

KDCSHUT NORMAL

In Anwendungen, die verteilte Transaktionsverarbeitung verwenden, sollte eine Anwendung immer mit KDCSHUT GRACE oder WARN beendet werden, denn dies erlaubt eine geordnete Beendigung von offenen verteilten Transaktionen.

Beim Beenden der Anwendung führt openUTM die folgenden Aktionen durch:

- Es werden alle Aufträge abgearbeitet, die noch in der UTM-Warteschlange sind.
- Die Verbindungen zu allen Kommunikationspartnern der Anwendung werden abgebaut.
- Die Verbindungen zu den Datenbanksystemen werden abgebaut.
- KDCFILE, System-Protokolldatei und Benutzer-Protokolldatei werden in einen konsistenten Zustand gebracht und ordnungsgemäß geschlossen.
- Alle Prozesse der Anwendung werden beendet.

Um eine UTM-Anwendung normal zu beenden, können Sie an Stelle des Kommandos KDCSHUT auch die entsprechende Funktion bei WinAdmin/WebAdmin oder an der Administrations-Programmschnittstelle verwenden.

Wird BCAM mit BCEND beendet, so wird eine zu diesem Zeitpunkt geladene UTM-Anwendung normal beendet.

Falls die UTM-Anwendung von einer Jobvariablen überwacht wird (siehe "Nach abnormalem Anwendungsende erneut starten"), so wird die erste Position der Jobvariablen auf „T“ und die zweite Position auf „N“ gesetzt.

> Mit dem Kommando /CANCEL-JOB lässt sich eine UTM-Task nur dann beenden, wenn sie im „nicht-privilegierten“ Zustand (TU) läuft. Mit diesem Kommando kann beispielsweise ein Teilprogramm in einer Endlos-Schleife beendet werden.

6.2 UTM-Anwendung abnormal beenden

Eine UTM-Anwendung kann durch folgende Ereignisse abnormal beendet werden:

- UTM-interner Fehler
- Fehler in der Systemumgebung, z.B. Datenbanksystem nicht verfügbar
- UTM-Administrationskommando KDCSHUT KILL (oder durch die entsprechende Funktion bei WinAdmin /WebAdmin oder KDCADMI)
- Generierungsfehler

Bei abnormaler Beendigung werden folgende Aktionen ausgeführt:

- Alle Transaktionen, die gerade von den einzelnen Prozessen bearbeitet werden, werden abgebrochen.
- Die Verbindungen zu allen Kommunikationspartnern der Anwendung werden abgebaut.
- Die Verbindung zum Datenbanksystem wird abgebaut.
- Für jede Task der Anwendung wird ein UTM-spezifischer Speicherauszug (UTM-Dump) erzeugt. Siehe dazu auch openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“.
- Alle Prozesse der Anwendung werden beendet und alle Dateien werden geschlossen. Es wird nicht versucht, die KDCFILE in einen konsistenten Zustand zu bringen. Dies geschieht erst bei einem erneuten Start der Anwendung.

Falls die UTM-Anwendung von einer Jobvariablen überwacht wurde (siehe "Nach abnormalem Anwendungsende erneut starten"), wird die erste Spalte der Jobvariablen auf „T“ und die zweite Spalte auf „A“ gesetzt.

Nach einem abnormalen Beenden der Anwendung sollten Sie als erstes die Ursache für den Abbruch feststellen. Dazu suchen Sie im SYSLST-Protokoll die Meldung K060. Diese Meldung enthält als Insert den Dump-Fehlercode, der genaue Auskunft über die Abbruch-Ursache gibt. Die Dump-Ursache können Sie auch als Teil des Dateinamens der UTM-Dumps finden. Die Bedeutung des Dump-Fehlercodes ist im openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“ bei der Meldung K060 beschrieben. Dabei gibt es drei Möglichkeiten:

- Der Dump-Fehlercode sagt aus, dass ein KDCDEF-Operand verändert werden muss. Dann ist die KDCFILE neu zu generieren. Wenn Sie dabei die Benutzerdaten im Pagepool erhalten wollen, gehen Sie wie folgt vor:
  - Warmstart mit ASYNTASKS=0, TASKS=1
  - Anwendung normal beenden mit KDCSHUT NORMAL
  - neue KDCDEF-Generierung mit dem veränderten Operanden
  - Übertragung der Benutzerdaten mit KDCUPD aus der alten in die neue KDCFILE
  - Start der Anwendung mit der neuen, aktualisierten KDCFILE

- Der Dump-Fehlercode nennt als Ursache
  - DMS-Fehler
  - Speicherengpass
  - Datenbanksystem ist z.Zt. nicht verfügbar

  Ist der Fehler behoben, können Sie die Anwendung erneut starten, openUTM führt dann automatisch einen Warmstart durch.
- Andernfalls (Dump-Fehlercode nicht beschrieben) liegt ein Systemfehler vor. In diesem Fall erstellen Sie Diagnoseunterlagen und schreiben eine Fehlermeldung an den Systembetreuer. Die Unterlagen, die der Systembetreuer benötigt, sind im Abschnitt „Diagnoseunterlagen für eine Problemmeldung“ zusammengestellt.

  Ein Warmstart mit derselben KDCFILE ist in diesem Fall nicht immer erfolgreich. Kann kein Warmstart durchgeführt werden, müssen Sie die KDCFILE mit KDCDEF neu generieren.

6.3 Diagnoseunterlagen für eine Problemmeldung

Beim abnormalen Ende einer UTM-Anwendung durch einen Systemfehler schreibt openUTM Dumps für alle Tasks der Anwendung, siehe openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“.

Folgende Diagnoseunterlagen sind unbedingt mitzuliefern, wenn Sie eine Problemmeldung für das Service Center schreiben:

- UTM-Dump-Dateien von allen Tasks
- SYSOUT-Protokolle von allen Tasks
- System-Protokolldatei SYSLOG
- KDCDEF-Steueranweisungen

UTM-Dumps und SYSLOG-Datei sollten nicht aufbereitet werden.

Zusätzlich sollten Sie für eventuelle Nachforderungen folgende Unterlagen sicherstellen:

- KDCFILE und bei Aufteilung auf mehrere Dateien zusätzlich alle Pagepool- und Wiederanlauf-Dateien
- das Protokoll vom KDCDEF-Lauf
- das Protokoll vom Binden des Anwendungsprogramms

7 UTM-Datenbank-Anwendung

Dieses Kapitel gibt einen zusammenhängenden Überblick über das, was für den Einsatz von Datenbanken unter openUTM zu beachten ist.

Folgende Systeme werden über die Schnittstelle zur UTM-DB-Zusammenarbeit (IUTMDB) angebunden:

- UDS/SQL
- SESAM/SQL
- LEASY (das Dateisystem LEASY verhält sich gegenüber openUTM wie ein Datenbank-System)
- CIS

Folgende Systeme werden über die XA-Schnittstelle angebunden:

- Oracle

Darüber hinaus kann openUTM auch mit anderen Datenbanksystemen koordiniert zusammenarbeiten, wenn diese Systeme entweder die IUTMDB- oder die XA-Schnittstelle unterstützen.

Multi-DB-Betrieb

openUTM kann auch koordiniert mit bis zu drei Datenbanksystemen unterschiedlichen Typs zusammenarbeiten, d. h. in einer UTM-Transaktion dürfen Aufrufe an mehrere Datenbanksysteme enthalten sein. Von den Datenbanksystemen, die Update-Operationen angefordert haben, darf nur eines kein „Two-Phase-Commit“-Protokoll unterstützen.

Für UDS/SQL und SESAM/SQL ist die koordinierte Zusammenarbeit von openUTM mit den beiden Systemen „in *einer* Multi-DB-Anwendung“ gewährleistet, wenn dabei folgende Schnittstellen benutzt werden:

- für Aufrufe an UDS/SQL die COBOL- oder Call-DML
- und für Aufrufe an SESAM/SQL die SQL.

Die Systeme UDS/SQL und SESAM/SQL können jedoch nicht in einer Anwendung kombiniert werden, falls für die Aufrufe an das DB-System UDS/SQL die SQL verwendet wird.

Auf Sonderfreigabe ist Multi-DB-Betrieb mit mehr als drei DB-Systemen möglich (maximal bis zu acht DB-Systeme).

Nähere Information über das Konzept der koordinierten Zusammenarbeit finden Sie im openUTM-Handbuch „Konzepte und Funktionen“.
Weiterführende Details sind in den Handbüchern zu den DB-Systemen unter dem Thema UTM bzw. openUTM beschrieben, z.B. „UDS/SQL - Anwendungen programmieren“ oder „SESAM/SQL - Basishandbuch“.

7.1 UTM-Datenbank-Anschluss generieren

Für eine koordinierte Zusammenarbeit müssen Sie den UTM-Datenbank-Anschluss in der KDCDEF-Anweisung DATABASE generieren. Der XA-Anschluss auf dem BS2000-System muss auch über die DATABASE-Anweisung generiert werden. Dort geben Sie an:

- den Typ des Datenbanksystems

  Für fremde Datenbanksysteme muss der Typ „DB“ angegeben werden.
- den Entry-Namen der Datenbank
- die Bibliothek, aus der das Datenbank-Verbindungsmodul nachgeladen werden kann.

  Für die Bibliothek können Sie einen symbolischen Linknamen angeben. Dadurch wird bei den Datenbanksystemen SESAM/SQL und UDS/SQL das Verbindungsmodul im IMON-Installationspfad gesucht und aus der Bibliothek nachgeladen. Dies hat den Vorteil, dass die UTM-Anwendung unabhängig ist von Installationskennungen und Bibliotheksnamen des Datenbanksystems.
- Zugangsdaten für die Datenbank (Username, Passwort).

  Diese Angaben sind optional und nur für Oracle-Datenbanken erlaubt. Wenn Sie die Zugangsdaten in der UTM-Generierung hinterlegen wollen, dann müssen Sie im Openstring für den Benutzernamen und das Passwort Platzhalter vorsehen.

Die Anweisung DATABASE kann mehrfach und in beliebiger Reihenfolge angegeben werden. Wenn Sie mehrere Datenbanksysteme vom gleichen Typ definieren, dann müssen sich die Entry-Namen unterscheiden. Dabei können alle Datenbanksysteme kombiniert werden, die man bei der UTM-Generierung angeben kann; eines der Systeme darf auch ein Fremdsystem sein (DATABASE=DB).

Näheres finden Sie im openUTM-Handbuch „Anwendungen generieren“ bei der Beschreibung der DATABASE-Anweisung.

ROOT-Source erzeugen und assemblieren

Beim KDCDEF-Lauf wird eine ROOT-Source erzeugt, die entsprechend den Angaben bei DATABASE folgende Makroaufrufe enthält:

- Bei Betrieb mit einem Datenbanksystem der Makroaufruf KDCDB.
- Bei Betrieb mit mehreren Datenbanksystemen die Makroaufrufe KDCDB*x*. Dabei ist*x* der erste Buchstabe des in DATABASE angegebenen Datenbank-Typs, z.B. KDCDBS für SESAM.

Das Makro KDCDB bzw. die Makros KDCDBx werden mit den jeweiligen Datenbanksystemen ausgeliefert. Beim Assemblieren der ROOT-Source müssen Sie die Bibliotheken, in der diese Makros zu finden sind, explizit zuweisen.

Näheres zum Erzeugen der Root-Source finden Sie im openUTM-Handbuch „Anwendungen generieren“ im Kapitel „Anwendungskomponenten generieren“.

7.2 UTM-Datenbank-Anwendung binden

openUTM benötigt für die koordinierte Zusammenarbeit mit einer Datenbank ein Datenbank-Verbindungsmodul. Dieses Modul wird mit dem betreffenden Datenbanksystem ausgeliefert. Die Bibliothek, in der dieses Verbindungsmodul steht, ist eine optionale Angabe in der KDCDEF-Anweisung DATABASE (siehe oben).

Das Verbindungsmodul kann wahlweise dynamisch nachgeladen werden oder beim Binden der Anwendung mit eingebunden werden. Wenn openUTM mit zwei Datenbanksystemen zusammenarbeiten soll, wird der für jedes Datenbanksystem spezifische Verbindungsmodul benötigt.

Verbindungsmodul dynamisch nachladen

Der Verweis auf den Verbindungsmodul kann beim Binden des Anwendungsprogramms weggelassen werden. Dies hat den Vorteil, dass das Anwendungsprogramm beim Wechsel auf eine neue Datenbankversion (mit neuem Verbindungsmodul) nicht neu gebunden werden muss. In diesem Fall wird das Verbindungsmodul beim Start der Anwendung dynamisch nachgeladen.

Dabei gilt der folgende Suchalgorithmus:

1. im Link-Kontext
2. im Shared Code des Benutzers
3. in nicht-privilegierten Subsystemen (z.B. kann das DB-System als Subsystem geladen werden)
4. im Shared Code des Systemadressraums
5. in der Bibliothek, die beim Generieren in der DATABASE-Anweisung angegeben wurde
6. in den alternativen Bibliotheken, die Sie beim Anwendungsstart mit einem SET-FILE-LINK-Kommando über die Linknamen BLSLIB*nn* (00<=*nn*<=99) bekanntgemacht haben.

Das heißt insbesondere, dass das Verbindungsmodul **nicht** aus der in der DATABASE-Anweisung angegebenen Bibliothek geladen wird, falls das DB-System vollständig als nicht-privilegiertes Subsystem geladen wurde oder falls Teile des DB-Systems als nicht-privilegierte Subsysteme geladen wurden und der Verbindungsmodul in einem dieser Teile enthalten ist.

> **ACHTUNG!**
>
> Beim Nachladen ist zu beachten, dass das Verbindungsmodul des Datenbanksystems gegebenenfalls nur im 24-Bit-Modus ablauffähig ist. In diesem Fall muss beim Binden der UTM-Anwendung die Anweisung
>
> MODIFY-SYMBOL-ATTRIBUTES ... ADDRESSING-MODE=24 eingefügt werden.

Verbindungsmodul einbinden

Wenn das Verbindungsmodul mit eingebunden wird, dann sollte man die INCLUDE-MODULES-Anweisung verwenden. Die RESOLVE-BY-AUTOLINK-Anweisung kann zu unerwünschten Nebeneffekten führen, wenn in den bei RESOLVE-BY-AUTOLINK angegebenen Bibliotheken außer dem Verbindungsmodul noch andere Module enthalten sind.

7.3 UTM-Datenbank-Anwendung starten und beenden

Eine UTM-Datenbank-Anwendung wird wie eine UTM-Anwendung ohne Datenbank gestartet und beendet, d.h. durch Starten und Beenden des UTM-Anwendungsprogramms.

Dabei sollten Sie beachten, dass der DBH (DatabaseHandler) der Datenbank immer **vor** der UTM-Anwendung gestartet wird, da es andernfalls einen Startfehler bei openUTM geben kann.

7.3.1 Startparameter für eine UTM-Datenbank-Anwendung

Für den Start einer UTM-DB-Anwendung müssen neben den Startparametern für openUTM auch die Startparameter für die Datenbank(en) angegeben werden, falls diese das fordern. Diese Startparameter sind Datenbank-spezifisch und werden in den Handbüchern des jeweiligen DB-Systems beschrieben.

Die Startparameter müssen nach folgendem Schema eingegeben werden, wobei die Startparameter von openUTM mit einer END-Anweisung abgeschlossen werden müssen und immer **vor** den Startparametern der Datenbank stehen:

`.UTM`	Startparameter für UTM, siehe Abschnitt „Startparameter der Anwendung“; eventuell muss der Parameter DBKEY angegeben werden.
	`...`
`.` *db-typ1*	Startparameter für die Datenbank vom Typ *db-typ1*, siehe Handbuch für das DB-System *db-typ1*.
`.` *db-typ2*	Startparameter für die Datenbank vom Typ *db-typ2*, wenn ein zweites DB-System eingesetzt wird, siehe Handbuch für das DB-System *db-typ2*.
`.` *db-typ3*	Startparameter für die Datenbank vom Typ *db-typ3*, wenn ein drittes DB-System eingesetzt wird, siehe Handbuch für das DB-System *db-typ3*.
	...

Für *db-typ1*, *db-typ2, db-typ3* müssen Sie den Datenbank-Typ angeben, den Sie in der DATABASE-Anweisung generiert haben.

Für UDS können Sie z.B. angeben:

```
.UDS DATABASE=UDSCONF
```

Damit wird der Anschluss der UTM-Anwendung an die UDS-Konfiguration UDSCONF festgelegt.

> ¡ Bei SESAM/SQL müssen die Startparameter in einer Konfigurationsdatei bereitgestellt werden, die den Tasks der UTM-SESAM-Anwendung unter dem Linknamen SESCONF zugewiesen wird.

7.3.2 Startparameter für eine UTM-Datenbank-Anwendung mit XA-Unterstützung

openUTM unterstützt auch auf BS2000-Systemen die XA-Funktionalität in der Kopplung mit Datenbanksystemen. Die Funktionalität im Betrieb mit herkömmlichen Datenbank-Systemen auf BS2000-Systemen ist davon nicht betroffen (z.B. UDS/SQL, SESAM/SQL oder LEASY).

Die Startparameter haben im allgemeinen folgendes Format:

```
.RMXA RM="...",OS="openstring",CS="closestring"
```

openstring Das Format von *openstring* ist Datenbank-spezifisch und in der Dokumentation des Datenbank-Systems beschrieben. Dort finden Sie auch die Bedeutung der einzelnen Parameter innerhalb des Openstring. openUTM übergibt die Strings aus der Startparameterdatei an das Datenbank-System, ohne sie zu überprüfen.

Die Startparameter für z.B. eine UTM-Oracle-Anwendung haben folgendes Format:

```
.RMXA RM="Oracle_XA",OS="openstring"
```

Für die Failover-Unterstützung des Oracle® Real Application Clusters sind weitere Parameter notwendig, siehe "Startparameter für Failover mit Oracle® Real Application Clusters".

Zugangsdaten für die Datenbank angeben

Sie können die Zugangsdaten für die Datenbank (Benutzername, Passwort) wahlweise in den Startparametern oder in der UTM-Generierung angeben, wobei Folgendes gilt:

- Die Angabe in den Startparametern hat Vorrang vor der UTM-Generierung. Sie können sowohl beides (Benutzernamen und Passwort) als auch nur das Passwort oder nur den Benutzernamen angeben. In allen drei Fällen wird beim Start der UTM-Anwendung die Meldung K238 ausgegeben.
- Wenn die Zugangsdaten aus der UTM-Generierung verwendet werden sollen (siehe "Oracle-Benutzername und Oracle-Passwort aus der UTM-Generierung verwenden"), dann müssen Sie in den Startparametern die Platzhalter *UTMUSER (für den Benutzernamen) und *UTMPASS (für das Passwort) angeben. Beim Verbindungsaufbau ersetzt openUTM automatisch die generischen Platzhalter *UTMUSER und *UTMPASS durch den Benutzernamen und das Passwort, die in der DATABASE-Anweisung generiert wurden.

Aus Sicherheitsgründen wird empfohlen, die Zugangsdaten in der UTM-Generierung zu hinterlegen.

Die Zugangsdaten lassen sich später per Administration ändern.

Beispiel für den Anschluss einer Oracle-XA-Datenbank

Startparameter:

```
.RMXA RM="Oracle_XA",OS="Oracle_XA+Acc=P/*UTMUSER
/*UTMPASS+SqlNet=RACSERVICE1+SesTm=60+DbgFl=0"
```

Passende KDCDEF-Generierung:

```
DATABASE TYPE=XA,ENTRY=XAOSWD,USERID='MaxTheSuperman',PASSWORD='PasswordOfMax'
```

7.3.2.1 Mehrere Instanzen

Bei Kopplung über die XA-Schnittstelle kann die UTM-Anwendung mehrere Oracle-Instanzen (Datenbanken) betreiben. Dazu müssen Sie für jede Instanz einen eigenen Openstring angeben. Jeder Openstring ist in einer eigenen Zeile in der Startparameterdatei anzugeben. In den Openstrings werden die verschiedenen Instanzen angegeben.

Die allgemeine Syntax für Startparameter von Oracle-XA-Instanzen ist:

```
.RMXA[xa-inst-name1] RM="Oracle_XA",OS="Oracle_XA[+DB=db-name1] ..."
.RMXAxa-inst-name2 RM="Oracle_XA",OS="Oracle_XA+DB=db-name2 ..."
.RMXAxa-inst-name3 RM="Oracle_XA",OS="Oracle_XA+DB=db-name3 ..."
```

Dabei muss der optionale Wert von *xa-inst-name1*, *xa-inst-name2*, ... jeweils zu dem Wert passen, der in der KDCDEF-Generierung bei DATABASE XA-INST-NAME= angegeben wurde.

Um in jedem Fall die Eindeutigkeit der Zuordnung von XA-Instanzen zur UTM-Generierungsinformation sicherzustellen, beachten Sie folgende drei Regeln:

- Verwenden Sie pro UTM-Anwendung maximal einen leeren XA-Instanz-Name und maximal einen leeren Oracle-DB-Namen.
- Alle XA-Instanz-Namen müssen in der UTM-Anwendung eindeutig sein.
- Alle Oracle-DB-Namen müssen in der UTM-Anwendung eindeutig sein.

Beispiel

Dieses Beispiel zeigt den Anschluss zweier Oracle-XA-Datenbanken an openUTM (BS2000).

Startparameter:

```
.RMXA RM="Oracle_XA",OS="Oracle_XA+Acc=P/*UTMUSER/
*UTMPASS+SqlNet=RACSRV1+SesTm=60+DbgFl=0"
.RMXAEVE RM="Oracle_XA",OS="Oracle_XA+DB=EVESDB+Acc=P/*UTMUSER/
*UTMPASS+SqlNet=RACSRV2+SesTm=60+DbgFl=0"
```

Der erste Startparameter enthält nur das Präfix „.RMXA", d.h. einen leeren XA-Instanznamen, und wird deshalb der ersten bei openUTM generierten Datenbank ohne XA-INST-NAME-Parameter zugeordnet.

Der zweite Startparameter enthält das Präfix „.RMXAEVE ", d.h. die zweite XA-Instanz wird der generierten Datenbank mit XA-INST-NAME=EVE zugeordnet, und die dafür generierten Zugangsdaten („Eve", ...) werden von dieser XA Instanz verwendet.

KDCDEF-Generierung:

```
DATABASE TYPE=XA,ENTRY=XAOSWD,USERID='MaxTheSuperman',PASSWORD='PasswordOfMax'
DATABASE TYPE=XA,ENTRY=XAOSWD,USERID='Eve',PASSWORD='PasswordOfEve',XA-INST-NAME=EVE
```

7.3.2.2 Oracle-Benutzername und Oracle-Passwort aus der UTM-Generierung verwenden

Die Zugangsberechtigung für eine Oracle Datenbank sollte aus Sicherheitsgründen per KDCDEF-Generierung festgelegt werden.

Beachten Sie bitte Folgendes:

- Der Oracle-Benutzername für die Verbindung zu Oracle und das dazugehörige Oracle-Passwort müssen in KDCDEF generiert sein (KDCDEF-Anweisung DATABASE, Operanden USERID und PASSWORD).

  Das Oracle-Passwort wird als Hashcode in den UTM-Systemtabellen abgespeichert (maskiert) und ist im UTM-Dump deshalb nicht im Klartext enthalten.
- Im Openstring des Startparameters geben Sie anstelle des Oracle-Benutzernamens den Platzhalter *UTMUSER und anstelle des Oracle-Passworts den Platzhalter *UTMPASS an. Diese Platzhalter werden nach folgenden Regeln ersetzt:
  - Enthält der Openstring mindestens einen der Platzhalter *UTMUSER oder *UTMPASS, so ersetzt UTM beim xa_open() Aufruf die Platzhalter durch die Werte, die für dieses Datenbanksystem generiert sind. D.h. im Openstring wird *UTMUSER durch den generierten Oracle-Benutzernamen und *UTMPASS durch das generierte Oracle-Passwort ersetzt.

    Aus Sicherheitsgründen wird das Oracle-Passwort erst unmittelbar vor der Verwendung beim xa_open() Aufruf in Klartext umgewandelt und sofort nach dem xa_open() Aufruf wieder im Prozessspeicher gelöscht.
  - Wenn der Openstring des Startparameters weder *UTMUSER noch *UTMPASS enthält, wird der Openstring unverändert an den xa_open() Aufruf übergeben.

Beachten Sie bitte, dass die Groß-/Kleinschreibung signifikant ist!

Beispiel

Sie möchten den Oracle-Benutzernamen und das Oracle-Passwort aus UTM-Generierung verwenden:

```
OS="Oracle_XA+SqlNet=O11+ACC=P/*UTMUSER/*UTMPASS+DbgFl=15"
```

Verhalten bei nicht generierten Oracle-Zugangsdaten

- Wurden die Operanden USERID und PASSWORD bei der UTM-Generierung nicht angegeben und verlangt die Oracle-Datenbank einen Benutzernamen und/oder ein Passwort, dann können Sie dies in den Startparametern angeben.
- Falls Sie im Startparameter *UTMUSER bzw. *UTMPASS angeben, obwohl die Operanden USERID und PASSWORD bei der UTM-Generierung nicht angegeben wurden, dann verwendet UTM einen leeren Oracle-Benutzernamen bzw. ein leeres Oracle-Passwort. D.h. der Verbindungsaufbau zur Datenbank wird in der Regel nicht erfolgreich sein.

7.3.2.3 Startparameter für Failover mit Oracle® Real Application Clusters

Eine UTM-Anwendung kommuniziert mit Oracle® Real Application Clusters über die XA-Schnittstelle. Kann in einem Failover-Fall ein XA-Aufruf von Oracle nicht mehr ordnungsgemäß durchgeführt werden, quittiert dies Oracle mit dem Rückgabewert „XAER_RMFAIL“.
Im Normalfall, d.h. wenn die Failover-Unterstützung nicht eingeschaltet ist, schließt openUTM aus dieser Meldung, dass eine weitere Zusammenarbeit mit dieser Datenbank nicht mehr möglich ist und bricht den Anwendungslauf ab. Um diesen Abbruch zu verhindern, geben Sie bei den Startparametern unter .RMXA zusätzlich den Wert RAC=Y an und steuern das Failover-Verhalten mit den optionalen Parametern RAC_retry und RAC_recover_down:

```
.RMXA RM="Oracle_XA",OS="openstring" ,RAC=Y[,RAC_retry=nnn]
                                      [,RAC_recover_down={Y|N}]
```

RAC=Y

schaltet die Failover-Unterstützung bei der Kopplung der UTM-Anwendung mit Oracle® Real Application Clusters ein.

RAC=N

schaltet die Failover-Unterstützung aus.
Standard: N

RAC_retry=nnn

nnn gibt die Anzahl der Versuche an, die openUTM unternehmen soll, um sich erneut mit der Datenbank zu verbinden, um einen einen Recover-Auftrag auszuführen.
Konnte für eine Transaktion im Zustand „Prepare-to-Commit“ der Commit-Auftrag wegen eines Failover nicht ausgeführt werden, verbindet sich openUTM erneut mit der Datenbank und führt einen Recover-Auftrag aus. Ist die aktuelle XID in der Liste der gelieferten XIDs enthalten, wird von openUTM ein Commit-Auftrag für die XID, also für die aktuelle Transaktion, ausgeführt. Ist die XID nicht in der Liste enthalten, wird von openUTM ein *xa_close* durchgeführt. Anschließend versucht openUTM erneut, sich mit der Datenbank zu verbinden, um einen Recover-Auftrag auszuführen.
Standard: 1

RAC_recover_down=

Legt fest, wie sich openUTM verhält, wenn die Transaktion nach der mit RAC_retry= festgelegten Anzahl von Versuchen nicht endgültig abgeschlossen werden konnte, d.h. nicht in den Zustand „Commit“ versetzt werden konnte.

N openUTM geht davon aus, dass die Transaktion bei Oracle® Real Application Clusters nicht mehr bekannt ist. Die Transaktion wird als „Commit“ angenommen und openUTM setzt den Anwendungslauf fort.
Standard: N

Y openUTM beendet den Anwendungslauf abnormal und erzwingt damit einen Warmstart, um für Datenkonsistenz zu sorgen.

Verhalten von openUTM im Failover-Fall

Wenn Sie Failover-Unterstützung eingeschaltet haben, dann verhalten sich openUTM und das Datenbanksystem wie folgt:

- Der Anwendungsabbruch wird vermieden, wenn ein Failover zu einem noch aktivenKnoten des Oracle® Real Application Clusters möglich ist.
- Bei Verbindungsverlust am Transaktionsende zwischen „Prepare“ und „Commit“ wird ein „Reconnect“ mit Recovery durchgeführt und im Erfolgsfall der „Commit“ über diese neue Verbindung wiederholt.
- Wenn zum Failover-Zeitpunkt noch Transaktionen offen sind, dann kann dies trotz eingeschalteter Failover-Unterstützung zu Problemsituationen und entsprechenden Fehlermeldungen führen (z.B. den Returncode ORA-25402 - transaction must rollback). Dies liegt daran, dass Oracle® Real Application Clusters beim Failover-Fall keine offenen Transaktionen migrieren kann. Diese Transaktionen müssen vom UTM-Anwendungsprogramm zurückgesetzt werden, siehe auch „UnterbrocheneTransaktionen“.

  Offene Mehrschritt-Transaktionen (d.h. nach PEND KP) werden im Failover-Fall durch das Datenbank-System zurückgesetzt, ohne dass openUTM dies beeinflussen kann.

  Das Datenbank-System wird nach dem Rollback automatisch neu verbunden; anschließend können wieder neue Transaktionen gestartet werden.
- Wenn der Failover-Fall während des Warmstarts der Anwendung oder der Beendigung eines UTM-Prozesses eintritt, wird die Fehlerbehandlung wie gewohnt durchgeführt, es wird dann kein „Reconnect“ versucht.
- Die Datenbank-Funktionalität „prepared Statements“ kann im Failover-Fall zu Fehlern führen.
- Der „Reconnect“ zum Datenbank-System kann durch Meldungen verfolgt werden.
  - xa_close im Reconnect-Fall:

    In der Meldung K202 wird im Insert &RMSTAT anstelle von „closed“ für die Instanzvon Oracle® Real Application Clusters der String „RAC closed“ ausgegeben.
  - xa_open im Reconnect-Fall:

    In der Meldung K224 wird im Insert &XACALL der String „RAC: xa_open“ ausgegeben.

  Debug-Meldungen

  In den Debug-Meldungen wird vermerkt, ob sich die Meldung auf eine Instanz von

Oracle® Real Application Clusters bezieht. Wie Sie XA-DEBUG-Informationen für den Anschluss an die Datenbank bekommen, ist in Abschnitt „Debug-Parameter“ beschrieben.

Unterbrochene Transaktionen

Unterbrochene Transaktionen können nur von dem Knoten fortgesetzt werden, der die Transaktion gestartet hat. Daher müssen alle UTM-Prozesse stets mit dem gleichen Knoten des Oracle® Real Application Clusters verbunden sein. Deshalb ist es am einfachsten,

- die UTM-Anwendung nach dem Failover des Oracle® Real Application Clusters zu beenden, bevor der ausgefallene Knoten neu gestartet wird,
- nach dem Neustart des ausgefallenen Knotens die UTM-Anwendung neu zu starten.

Dadurch ist sichergestellt, dass alle UTM-Prozesse mit dem gleichen Knoten des Oracle® Real Application Clusters verbunden sind und alle Transaktionen der Anwendung vom neu gestarteten Knoten des Oracle® Real Application Clusters bearbeitet werden.

Falls ein Beenden und Neustart der UTM-Anwendung nicht möglich ist, d.h. falls Knoten des Oracle® Real Application Clusters bei laufender UTM-Anwendung umgeschaltet werden, dann kann sich folgende Situation ergeben, in der nicht mehr alle UTM-Prozesse mit demselben Knoten verbunden sind:

- Eine Transaktion wird durch das Failover unterbrochen; zu diesem Zeitpunkt ist der UTM-Prozess noch mit dem alten Knoten verbunden.
- Durch Nachstarten des Prozesses oder nach einem PEND ER im UTM-Anwendungsprogramm wird die unterbrochene Transaktion von einem anderen UTM-Prozess weitergeführt. Dieser Prozess ist jetzt mit dem neuen Knoten verbunden.
- Die Datenbank-Instanz lehnt die Wiederaufnahme der unterbrochenen Transaktion ab (xa-start mit RESUME) und meldet, dass die Transaktion nicht bekannt ist.
- openUTM führt einen „Reconnect“ zur Datenbank-Instanz aus. Auf der neuen Verbindung (d.h. mit dem neuen Knoten) versucht openUTM, die Transaktion wieder aufzunehmen.
- Das Datenbank-System lehnt dies erneut ab, da die Datenbank-Transaktion auf demalten Knoten des Oracle® Real Application Clusters begonnen wurde und auf dem neuen nicht fortgesetzt werden kann.
- openUTM setzt die globale Transaktion zurück und gibt eine K160-Meldung aus, im Insert des internen Returncodes KCRCDC wird „NOTA“ ausgegeben.

Eine solche Situation lässt sich wie folgt mit Hilfe des MSGTAC-Programms behandeln.

Steuerung über das MSGTAC-Programm

Für die K160-Meldung wird als zusätzliches Meldungsziel der Event-Service MSGTAC definiert. MSGTAC reagiert auf den Insert der Meldung und stößt über die Programmschnittstelle zur Administration (KC_CHANGE_APPLICATION) ein Neustarten an. Das Anwendungsprogramm wird neu geladen und wird anschließend mit dem neuen Knoten verbunden.
Durch dieses Verfahren wird die Zeitspanne minimiert, in der die UTM-Prozesse mit unterschiedlichen Knoten verbunden sind. Die Zahl der zurückgesetzten Transaktionen beschränkt sich auf diejenigen, die auf dem alten Knoten begonnen wurden und auf dem neuen nicht fortgesetzt werden konnten. Die Transaktionen, die auf dem neuen Knoten vor dem Neustarten begonnen wurden, können weitergeführt werden.

Oracle®-Anschluss

Die Verbindung zu einer Oracle®-Datenbank wird über einen sogenannten „Service“ aufgebaut. In einer Oracle® Real Application Clusters-Umgebung können Sie zusätzlich "DTP services" einrichten.

Daraus ergeben sich für den laufenden Betrieb folgende Möglichkeiten:

- automatische Fehlererkennung
- automatisches Failover:
  Nach Ausfall einer Instanz wird eine neue Transaktion auf eine andere Instanz des „service“ umgeleitet. Es muss nicht administrativ eingegriffen werden.
- Lastverteilung bereits beim Verbindungsaufbau

DTP service anlegen (Oracle®)

1. Richten Sie mit dem Kommando “srvctl add service“ einen neuen "service" für die Datenbank ein und ordnen Sie ihn einer Instanz der Datenbank zu.

   Beispiel:

   Für die RAC-Datenbank `dbracutm` mit den Instanzen `racutm1` und `racutm2` sollen zwei "DTP services" mit folgenden Optionen angelegt werden:

-d	Name der Datenbank
-s	Name des (DTP-)Services
-r	Name der Erst-Instanz
-a	Name der Zweit-Instanz
-P	Methode des Failovers

   ```
   "srvctl add service -d dbracutm -s racutmS12 -r racutm1
                                                -a racutm2
                                                -P BASIC"
   ```

   und

   ```
   "srvctl add service -d dbracutm -s racutmS21 -r racutm2
                                                -a racutm1
                                                -P BASIC"
   ```

   Der Service `racutmS12` verbindet sich mit der Instanz `racutm1` und im Failover-Fall mit Instanz `racutm2`. Umgekehrt verbindet sich Service `racutmS21` mit der Instanz `racutm2` und im Failover-Fall mit Instanz `racutm1`.

2. Wandeln Sie mit SQLPLUS die Services in "DTP services“ um:

   ```
   SQL> connect ....
   SQL> execute dbms_service.modify_service
                ( service_name => 'racutmS12', dtp => true );
   SQL> execute dbms_service.modify_service
                ( service_name => 'racutmS21', dtp => true );
   ```

   ```
   SQL> exit
   ```

   Sie können die (DTP-)Services mit "srvctl-Kommandos" starten, stoppen und administrieren, siehe auch Oracle®-Handbuch "Administration and Deployment Guide".

> **i** Der DTP-Service muss auf dem Knoten gestartet werden, auf dem die ihm primär zugewiesene Instanz des RAC-DB-Systems läuft, d.h. der DTP-Service `racutmS21`, der primär der Instanz `racutm2` zugeordnet ist, muss auf dem Knoten gestartet werden, auf dem diese Instanz läuft.

1. Tragen Sie den "service" in der Datei `tnsnames.ora` mit einem net_service_name ein:

   Beispiel

   ```
   RACUTMS1 =
     (DESCRIPTION =
        (ADDRESS_LIST =
           (ADDRESS = (PROTOCOL = TCP) (HOST=server1) (PORT=1521))
           (ADDRESS = (PROTOCOL = TCP) (HOST=server2) (PORT=1521))
        )
        (CONNECT_DATA =
           (SERVICE_NAME = racutmS12.domain_name )
        )
        (FAIL_OVER = ON)
     )
   ```

2. Weisen Sie in den Startparametern im Open-String dem Operanden "SqlNet" diesen net_service_name (hier RACUTMS1) zu.

7.3.2.4 Debug-Parameter

Sie haben die Möglichkeit, zu Testzwecken die XA-Schnittstelle in openUTM zu protokollieren. Dazu steht Ihnen der RMXA-Startparameter DEBUG= zur Verfügung. Dieser Parameter muss **vor** den anderen RMXA-Startparametern angegeben werden, er wird nicht an das Datenbank-System weitergereicht.

> ¡ Es wird empfohlen, diese Funktionalität, nur zu Testzwecken zu verwenden, da die Ausgaben umfangreich werden können. Jede Transaktion wird in der Task protokolliert, unabhängig davon, ob ein Zugriff auf die Datenbank erfolgt.

Der Parameter DEBUG= hat folgendes Format:

```
.RMXA DEBUG={ YES | ALL },OUTPUT={ SYSOUT | FILE }
```

Erläuterung

DEBUG= Einschalten der Debug-Funktion.

Verwenden Sie DEBUG= als ersten RMXA-Startparameter.

YES protokolliert werden die einzelnen XA-Aufrufe, sowie für jeden Aufruf

- die Vorgangsnummer
- der Transaktionszähler
- der Rückgabewert

ALL protokolliert werden zusätzlich zu den Werten bei DEBUG=YES jeweils die Statuswerte und die XID.

OUTPUT= bestimmt das Ziel der Ausgabe

SYSOUT es wird auf SYSOUT protokolliert.

FILE die Ausgabe erfolgt in eine Datei. Der Dateiname hat folgendes Format:
KDC.TRC.XA.*appliname.tsn*

appliname
Name der Anwendung, maximal 8 Zeichen

tsn
TSN der UTM-Task (4-stellig)

Alternativ kann eine Datei über den Linknamen KDCXA zugewiesen werden.

Beispiel:

```
/SET-FILE-LINK LINK-NAME=KDCXA,FILE-NAME=XA-DEBUG.BSP.01
```

Die Protokollierung der XA-Schnittstelle können Sie während des Anwendungslaufs auch per Administration ein- oder ausschalten. Verwenden Sie dazu die Programmschnittstelle, die Administrationstools WinAdmin/WebAdmin oder das Administrationskommando KDCDIAG XA-DEBUG=. Details finden Sie im openUTM-Handbuch „Anwendungen administrieren“.

7.3.3 UTM-Datenbank-Anwendung beenden

UTM-Datenbank-Anwendung normal beenden

Eine UTM-Datenbank-Anwendung beenden Sie mit Hilfe der UTM-Administration, siehe Abschnitt „UTM-Anwendung per Administration normal beenden“. Der Database Handler wird dabei nicht heruntergefahren.

UTM-Datenbank-Anwendung abnormal beenden

Eine UTM-Datenbank-Anwendung kann per Administration oder aufgrund von Fehlern abnormal beendet werden, siehe Abschnitt „UTM-Anwendung abnormal beenden“. Nach einem abnormalen Anwendungsende kann es vorkommen, dass die Datenbank und KDCFILE in einem nicht-konsistentem Zustand sind.

In diesem Fall wird die Konsistenz der Daten beim anschließenden Warmstart der UTM-Datenbank-Anwendung überprüft und ggf. wieder hergestellt. Dabei führt openUTM eine gemeinsame Recovery Phase mit den beteiligten Datenbanksystemen durch.

7.4 Betrieb einer UTM-Datenbank-Anwendung

Der Betrieb einer UTM-Datenbank-Anwendung gehorcht den selben Prinzipien wie der Betrieb einer UTM-Anwendung ohne Datenbank. Die Besonderheiten, die dabei zu beachten sind, sind in den nachfolgenden Abschnitten beschrieben.

7.4.1 Anmelden und Abmelden eines Benutzers

Ein Benutzer, der mit einer UTM-Datenbank-Anwendung arbeiten möchte, meldet sich über die Client-spezifischen Anmeldeverfahren bei openUTM an. Entsprechendes gilt für das Abmelden.

Wenn zum Anmelden ein Anmelde-Vorgang benützt wird, dann ist Folgendes zu beachten:

- Meldet sich der Benutzer über ein Terminal oder eine Terminalemulation an, dann sind Datenbank-Aufrufe im ersten Teil des SIGNON-Services aus Sicherheitsgründen nicht erlaubt, es sei denn, man lässt dies per UTM-Generierung explizit zu mit der KDCDEF-Anweisung SIGNON, ...RESTRICTED=NO.
- Im zweiten Teil des Anmelde-Vorgangs kann das Berechtigungsprofil für den Benutzer aus der Datenbank gelesen werden, sofern die Datenbank diese Möglichkeit unterstützt. Damit lässt sich ein DB/DC-übergreifendes Berechtigungskonzept realisieren.

Näheres zum An- und Abmelden finden Sie im Kapitel „Arbeiten mit einer UTM-Anwendung“.

7.4.2 SAT-Protokollierung

Folgendes gilt nur für den DB-Anschluss über die IUTMDB-Schnittstelle, d.h. nicht für XA-Datenbanken wie z.B. Oracle.

Die SAT-Protokollierung kann für jeden Transaktionscode (= auftragsgesteuert) und jeden UTM-Benutzer (= benutzergesteuert) einzeln definiert werden. UTM protokolliert dann alle sicherheitsrelevanten Ereignisse innerhalb von UTM und setzt an der Schnittstelle zum DB-System eine Anzeige, dass die SAT-Protokollierung aktiviert wurde.

Sicherheitsrelevante Ereignisse innerhalb der DB-Transaktionen müssen vom DB-System protokolliert werden, falls dies vorgesehen ist. UTM protokolliert nur, ob die Transaktion als Ganzes erfolgreich ausgeführt wurde oder nicht.

Näheres zur SAT-Protokollierung finden Sie in Abschnitt „Benutzergesteuerte SAT-Protokollierung“ und in Abschnitt „Auftragsgesteuerte SAT-Protokollierung“.

7.4.3 Accounting

Folgendes gilt nur für den DB-Anschluss über die IUTMDB-Schnittstelle, d.h. nicht für XA-Datenbanken wie z.B. Oracle.

DB-Systeme, die nicht selbst in UTM-Tasks ablaufen, können openUTM den Betriebsmittelverbrauch für die Bearbeitung von DB-Aufrufen über die IUTMDB-Schnittstelle mitteilen. Die vom DB-System gelieferten Verbrauchswerte (CPU, I/Os) werden in der Kalkulationsphase im Kalkulationssatz abgelegt, siehe Abschnitt „Aufbau des Kalkulationssatzes“.

Diese Werte werden auch an openSM2 geliefert. Da diese Daten u.U. vom DB-System nicht exakt ermittelt werden, ist ihre Auswertung zu Messzwecken nur eingeschränkt möglich, Tendenzen sind jedoch feststellbar. Zum kurzzeitigen Erfassen exakter Werte vom DB-System steht der Messmonitor KDCMON zur Verfügung, siehe unten.

In der Abrechnungsphase werden sie mit den festgelegten Gewichten multipliziert und auf den Verrechnungseinheitenzähler addiert, siehe Abschnitt „Aufbau des Abrechnungssatzes“.

Die Verbrauchswerte werden von den DB-Systemen SESAM/SQL und UDS/SQL an openUTM geliefert. SESAM /SQL liefern die Verbrauchswerte nur dann an openUTM, wenn auch im DBH die Sammlung von Abrechnungsinformationen eingeschaltet ist, siehe dazu Handbuch „Datenbankbetrieb SESAM/SQL“.

7.4.4 Leistungskontrolle

DB-Aufrufe aus den Teilprogrammen können nur für die IUTMDB-Schnittstelle erfasst werden, d.h. nicht für XA-Datenbanken wie z.B. Oracle.

Zur Leistungskontrolle der UTM-Anwendung kann der UTM-Messmonitor KDCMON mit dem Aufbereitungs-Tool KDCEVAL eingesetzt werden. Einige der mit KDCEVAL aufbereiteten Listen enthalten auch Datenbank-spezifische Informationen:

- Die Liste TASKS enthält in der Spalte `database` den Zeitanteil für DB-Aufrufe.
- Die Liste KCOP gibt in NOOP an, wie oft DB-Messdaten geschrieben wurden.
- Die Liste TACLIST gibt TAC-spezifisch die Anzahl der DB-Aufrufe an.
- Die Listen TRACE und TRACE2 sind Trace-Listen, die sowohl die UTM-Aufrufe als auch die DB-Aufrufe enthalten.

Näheres zu KDCMON und den zugehörigen Auswertungslisten finden Sie im Abschnitt„UTM-Messmonitor KDCMON“

7.4.5 Diagnose

Wichtige Informationsquellen zur Fehlerdiagnose einer UTM-Anwendung sind UTM-Meldungen, Fehlercodes und Dumps. Bei einer UTM-Datenbank-Anwendung findet man darin zusätzlich Datenbank-spezifische Informationen. Diese sollten als erstes herangezogen werden, wenn ein Fehler mit der Datenbank-Kopplung zusammenhängen könnte. Dabei handelt es sich um folgende UTM-Diagnose-Informationen:

- Die Datenbank-spezifischen UTM-Meldungen K068 und K071
- Die Start-Fehlercodes der Meldung K049
- Meldungen des XA-Datenbankanschlusses K201 bis K233
- Den inkompatiblen Returncode KCRCDC
- Die DB Diagarea des UTM-Dumps, falls ein UTM-Dump erzeugt wurde
- Die CDUMP-Fehlercodes KDCDBxx, falls ein CDUMP erzeugt wurde

Einzelheiten dazu finden Sie im openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“.

8 Arbeiten mit einer UTM-Anwendung

Dieses Kapitel beschreibt, wie ein Benutzer über die unterschiedlichen Clients mit einer UTM-Anwendung kommunizieren kann. Für alle Clients verläuft die Kommunikation immer nach folgendem Prinzip:

1. Anmelden an die UTM-Anwendung
   Ein Benutzer kann sich nur über Clients anmelden, für die LTERM-Partner, LTERM-Pools oder OSI-LPAP-Partner in der UTM-Anwendung generiert sind, siehe openUTM-Handbuch „Anwendungen generieren“.
2. Services (=Vorgänge) der UTM-Anwendung aufrufen:
   Für den Zugriffsschutz bietet openUTM ein eigenes Berechtigungskonzept, siehe "Berechtigungskonzept von openUTM".
3. Ggf. UTM-Benutzerkommandos eingeben.
4. Abmelden von der UTM-Anwendung

Im Detail unterscheiden sich diese Schritte je nach Art des Clients. Die folgenden Abschnitte beschreiben, welche Möglichkeiten es für die verschiedenen Clients gibt.

Für den Zugang werden UTM-Benutzerkennungen verwendet, sofern die Anwendung mit Benutzerkennungen generiert ist. Das Anmelden an eine UTM-Anwendung ohne Benutzerkennungen wird im Kapitel "Anmeldeverfahren ohne Benutzerkennungen" beschrieben.

8.1 Anmeldeverfahren mit Benutzerkennungen

Ist eine Anwendung mit Benutzerkennungen generiert, dann führt openUTM für den Benutzer abhängig von der Art des Client ein Standard-Anmeldeverfahren durch. Für manche Arten von Clients besteht daneben die Möglichkeit, anstelle des Standard-Verfahrens selbst erstellte Anmeldeverfahren zu verwenden, siehe Abschnitt „Anmeldeverfahren mit Anmelde-Vorgängen“.

Ein Benutzer kann sich über folgende Client-Zugänge anmelden:

- Terminals ("Standard-Anmeldeverfahren für Terminals")
- UPIC-Clients und TS-Anwendungen ("Anmeldeverfahren für UPIC-Clients und TS-Anwendungen")
- OSI TP-Partner ("Anmeldeverfahren für OSI TP-Partner")
- HTTP-Clients ("Anmeldeverfahren für HTTP-Clients")
- über das Internet mit Hilfe von Web Services (WS4UTM) ("Anmeldeverfahren im World Wide Web über Web Services (WS4UTM)")
- über das Internet mit Hilfe von WebTransactions ("Anmeldeverfahren im World Wide Web über WebTransactions")

Unter bestimmten Voraussetzungen können sich auch mehrere Benutzer unter einer Benutzerkennung anmelden, siehe Abschnitt „Mehrfach-Anmeldungen unter einer Benutzerkennung“.

8.1.1 Standard-Anmeldeverfahren für Terminals

Der Benutzer führt folgende Schritte durch, um über ein Terminal mit einer UTM-Anwendung zu arbeiten:

1. Verbindungsaufbau zur UTM-Anwendung

   Die Initiative zum Verbindungsaufbau kann entweder von der UTM-Anwendung oder vom Benutzer ausgehen.
2. Anmelden bei einer UTM-Anwendung, siehe Abschnitt „Standard-Anmelde-Dialog“.

Erst dann kann der Benutzer Vorgänge starten und im Dialog bearbeiten, siehe Abschnitt „UTM-Services aufrufen“.

Verbindungsaufbau durch den Benutzer über Terminal-Emulation

Ein Benutzer kann auch von einem Windows-PC aus mit einer UTM-Anwendung arbeiten, indem er eine Emulation wie z.B. MT9750 aufruft und eine Sitzung öffnet, die für diese Anwendung konfiguriert ist.

Wenn Sie eine Sitzung für MT9750 einrichten, dann geben Sie den Namen der UTM-Anwendung sowie den Namen und/oder die IP-Adresse des Rechners an, auf dem die Anwendung läuft. Details dazu finden Sie im MT9750 Produkthandbuch.

8.1.1.1 Standard-Anmelde-Dialog

Der Standard-Anmelde-Dialog wird immer dann durchgeführt, wenn die beiden folgenden Bedingungen erfüllt sind:

1. Für das Terminal ist kein automatisches KDCSIGN (= automatische Berechtigungsprüfung) generiert (siehe "Automatisches KDCSIGN").
2. Für den Standard-Anwendungsnamen (generiert in MAX APPLINAME), ist kein Anmelde-Vorgang generiert (siehe "Anmeldeverfahren mit Anmelde-Vorgängen").

Beim Standard-Anmelde-Dialog führt openUTM eine Berechtigungsprüfung durch (Zugangskontrolle). Dabei wird der Benutzer über UTM-Meldungen im Zeilenmodus aufgefordert, sich zu legitimieren.

Man kann bei der UTM-Generierung zwischen verschiedenen Varianten wählen, auch die Meldungstexte können verändert werden. Darüber hinaus ist dieses Verfahren jedoch nicht modifizierbar.

Die Berechtigungsprüfung kann unterschiedlich streng festgelegt werden. Eine Übersicht aller Möglichkeiten enthält die Abbildung in Abschnitt „Szenarien für die UTM-Berechtigungsprüfung“.

openUTM fordert den Benutzer mit folgender Meldung zur Berechtigungsprüfung auf:

```
K002 Verbunden mit der Anwendung <anwendungsname> - Bitte KDCSIGN
```

Der Benutzer muss seine Berechtigung mit der folgenden Eingabe nachweisen:

```
KDCSIGN kennung [,password]
```

Im Kommando KDCSIGN kann *password* angegeben werden, wenn das Passwort für diesen Benutzer ohne Dunkelsteuerung generiert ist (KDCDEF-Anweisung USER...,PASS=).

openUTM prüft die Benutzerkennung und ggf. das Passwort. Ist das Ergebnis negativ, dann wird der Benutzer zur Wiederholung der KDCSIGN-Eingabe aufgefordert. Ist der Benutzer mit Passwort generiert, aber das Passwort wurde in dem Kommando KDCSIGN nicht angegeben, dann fordert openUTM das Passwort in einem zweiten Dialogschritt dunkel gesteuert an.

Dunkelsteuerung des Passworts

Wenn für die Benutzerkennung Dunkelsteuerung des Passworts generiert ist (USER...,PASS=(*password*,DARK)), dann fordert openUTM den Benutzer mit einer Meldung auf, das Passwort in ein dunkel gesteuertes Feld einzugeben.

Dabei hat der Benutzer die Möglichkeit, ein neues Passwort einzugeben, welches das bisherige ersetzen soll, vorausgesetzt, die in der Generierung festgelegte minimale Gültigkeitsdauer erlaubt die Passwortänderung zu diesem Zeitpunkt. Das neue Passwort ist hierbei durch eine Wiederholung zu bestätigen. openUTM prüft das angegebene alte Passwort und ggf. das neue Passwort. Ist das alte Passwort nicht richtig, dann wird der Benutzer zur Wiederholung der KDCSIGN-Angabe aufgefordert. Stimmen die Angaben zum neuen Passwort nicht überein, dann wird der Benutzer mit einer UTM-Meldung informiert und zur Wiederholung der Eingabe aufgefordert.

Gültigkeitsdauer des Passwortes

Bei der Generierung der Benutzerkennung kann eine maximale und eine minimale Gültigkeitsdauer für das Passwort vereinbart werden:

USER ...,PROTECT-PW=(...,*maxtime*,*mintime*)

Die minimale Gültigkeitsdauer bedeutet, dass der Benutzer nach einer Passwort-Änderung die nächste Änderung erst nach Ablauf der festgelegten Zeit vornehmen kann. Die maximale Gültigkeitsdauer bedeutet, dass der Benutzer das Passwort jeweils innerhalb der angegebenen Zeitspanne ändern muss.

Wird das Passwort innerhalb der nächsten 14 Tage nach der Anmeldung ungültig, warnt openUTM den Benutzer mit einer K-Meldung, sofern die minimale Gültigkeitsdauer des Passworts eine Änderung zu diesem Zeitpunkt erlaubt. Ein Passwort kann der Benutzer wie unter „Dunkelsteuerung des Passworts“ beschrieben ändern.
Um zu verhindern, dass Benutzer, die längere Zeit nicht mit der Anwendung arbeiten, die Passwort-Änderung versäumen und sich dann an den Administrator wenden müssen, kann die UTM-Anwendung so konfiguriert werden, dass auch nach Ablauf des Passworts eine Anmeldung erlaubt wird, siehe Abschnitt „Grace-Sign-On“.

openUTM überprüft bei der Passwortänderung, ob:

- das neue Passwort sich von dem alten Passwort unterscheidet, wenn eine maximale Passwort-Gültigkeitsdauer generiert ist.
  Wenn eine Passwort-Historie generiert ist (SIGNON ...,PW-HISTORY=*n*), dann wird das neue Passwort auch gegen die letzten *n* Passworte geprüft.
- das neue Passwort der Komplexitätsstufe entspricht, die für die Benutzerkennung generiert wurde (USER ..., PROTECT-PW=).
- die Länge des Passwortes größer oder gleich der generierten Mindestlänge ist (USER ...,PROTECT-PW=).

Erfüllt das neue Passwort alle diese Punkte, nimmt openUTM die Passwortänderung vor. Die Gültigkeitsdauer des neuen Passwortes entspricht wieder dem generierten Wert.
Erfüllt das neue Passwort einen dieser Punkte nicht, so wird der Benutzer mit der folgenden Meldung aufgefordert, die KDCSIGN-Eingabe mit dem alten Passwort zu wiederholen:

```
K097 Angaben zum neuen Passwort sind nicht verwendbar - Bitte KDCSIGN
```

Ist die Gültigkeitsdauer des Passwortes bei der Anmeldung bereits abgelaufen und ist kein Grace-Sign-On generiert, so wird die Anmeldung mit der folgenden Meldung zurückgewiesen:

```
K120 Die Gueltigkeit des Passworts ist abgelaufen - Bitte KDCSIGN
```

Eine Anmeldung an die UTM-Anwendung unter dieser Benutzerkennung ist erst dann wieder möglich, wenn der UTM-Administrator der Kennung ein neues Passwort zugewiesen hat.

Grace-Sign-On

Ist die Gültigkeitsdauer des Passwortes bei der Anmeldung bereits abgelaufen und ist die Anwendung mit Grace-Sign-On generiert (SIGNON ...,GRACE=YES), so wird der Benutzer mit einer K-Meldung darauf hingewiesen, dass sein Passwort nicht mehr gültig ist. Gleichzeitig wird er aufgefordert, sein bisheriges Passwort und ein neues Passwort einzugeben. Zum Ändern muss auch ein eventuell bereits eingegebenes Passwort nochmals in das entsprechende dunkelgesteuerte Feld eingegeben werden.

KDCSIGN mit Ausweiskarte

Wenn für die Benutzerkennung per UTM-Generierung das Einlegen einer Ausweiskarte (Magnetstreifenkarte) vorgeschrieben ist, fordert openUTM den Benutzer mit einer Meldung auf, eine Ausweiskarte in den Ausweisleser einzulegen. Zeitgleich zu dieser Meldung wird die Tastatur des Terminals gesperrt. Die Sperre wird erst durch Einlegen des Ausweises aufgehoben. Ist kein Ausweis verfügbar, so kann die Tastatur ersatzweise mit der Taste K14 oder ESC : entsperrt werden.

openUTM prüft die Ausweisinformation, d.h. openUTM vergleicht die in der KDCDEF-Anweisung USER ...,CARD=(...) definierte Zeichenfolge mit der entsprechenden Zeichenfolge auf dem Ausweis. Fällt das Ergebnis negativ aus, so wird der Benutzer mit der Meldung K031 informiert und zur Wiederholung der KDCSIGN-Eingabe aufgefordert. Fällt das Ergebnis positiv aus, dann kann der Benutzer mit der Anwendung arbeiten.

Der Ausweis muss solange im Ausweisleser bleiben, wie der Benutzer unter dieser Benutzerkennung arbeiten möchte. Wird er früher herausgezogen, so erkennt openUTM dies spätestens bei der nächsten Dialogeingabe und erzeugt implizit KDCOFF. Sehen Sie dazu auch Abschnitt „Abmelden von der UTM-Anwendung“.

> **i** Das Herausziehen des Ausweises erzeugt eine Nachricht entsprechend dem Drücken der Funktionstaste K14. Arbeitet der Benutzer unter einer Benutzerkennung, die KDCSIGN mit Ausweisprüfung verlangt, so ist die Zuordnung von K14 (KDCDEF-Anweisung SFUNC) zu einem TAC oder Returncode für diese Benutzerkennung nicht sinnvoll. Wird der Funktionstaste K14 das Kommando KDCOFF zugeweisen, so wirkt die Eingabe von K14 für alle Benutzerkennungen wie KDCOFF.

Szenarien für die UTM-Berechtigungsprüfung

Das folgende Diagramm zeigt, welche Varianten der UTM-Berechtigungsprüfung möglich sind - abhängig von der KDCDEF-Generierung. Bei fehlerhaften Eingaben gibt openUTM eine spezifische Meldung aus und fordert den Benutzer zu einer neuen Eingabe auf. Werden von einem bestimmten Terminal aus oder unter einer bestimmten Benutzerkennung nacheinander mehrere erfolglose Anmeldeversuche unternommen, dann erzeugt openUTM die Meldung K094 mit dem Standardziel SYSLOG (System-Protokolldatei). Die maximal zulässige Anzahl erfolgloser Anmeldeversuche bis zum Auslösen der Meldung K094 kann mit SIGNON ... SILENT-ALARM= bei der KDCDEF-Generierung festgelegt werden. Ein MSGTAC-Teilprogramm kann auf diese Meldung reagieren.

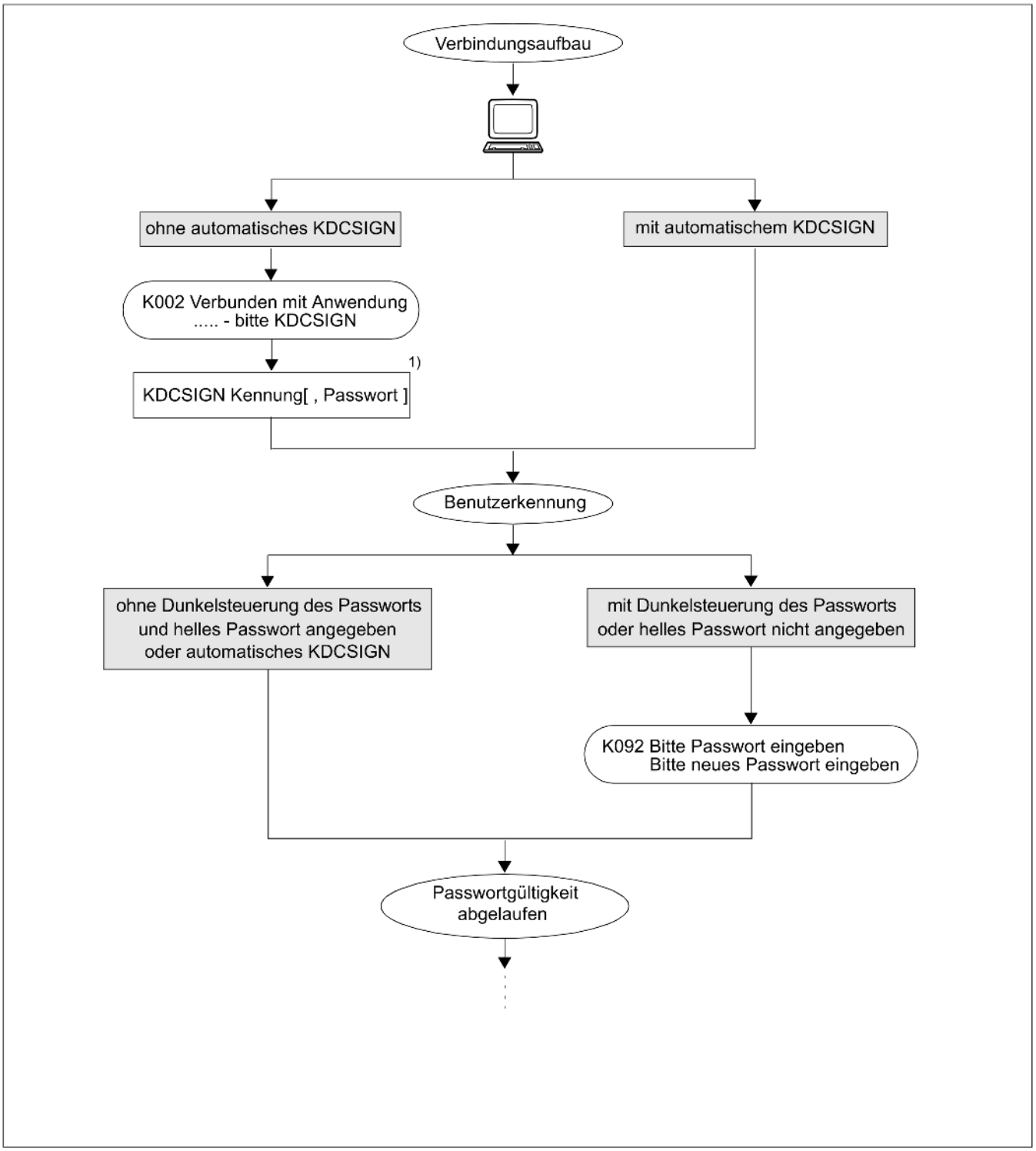


Bild 5: Szenarien der Berechtigungsprüfung bei Anwendungen mit Benutzerkennungen (Teil 1)

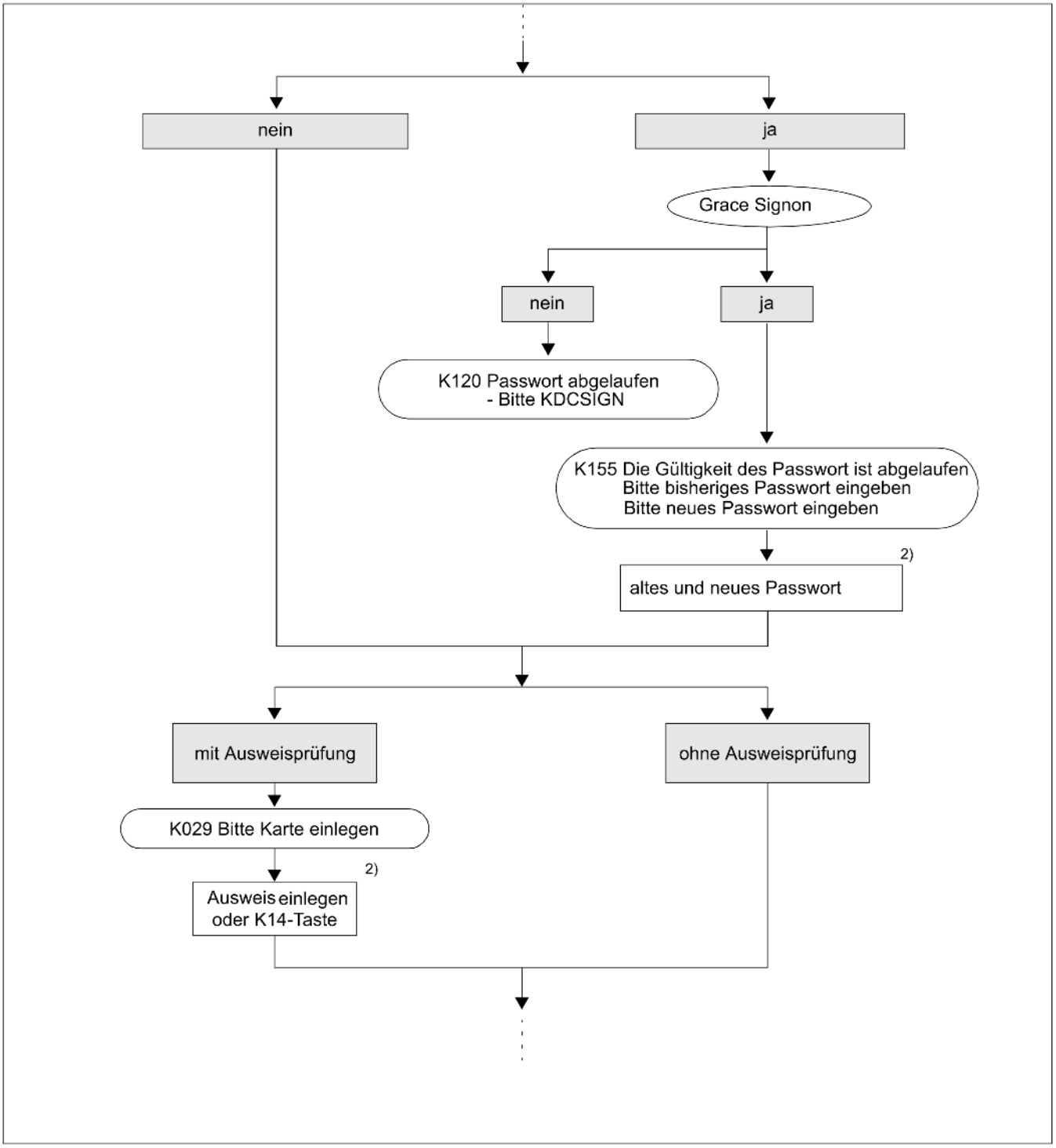


Bild 6: Szenarien der Berechtigungsprüfung bei Anwendungen mit Benutzerkennungen (Teil 2)

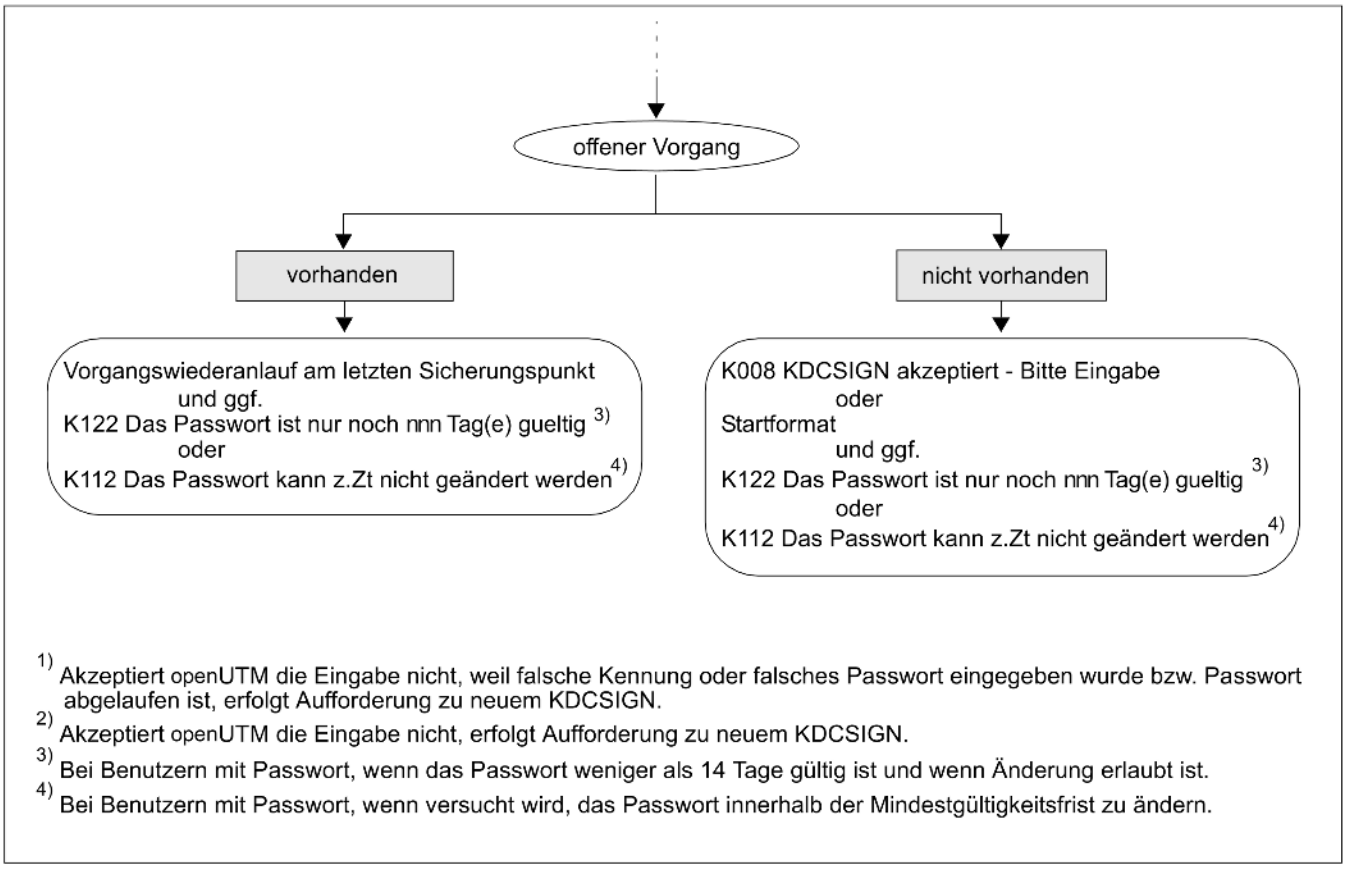


Bild 7: Szenarien der Berechtigungsprüfung bei Anwendungen mit Benutzerkennungen (Teil 3)

8.1.1.2 Automatisches KDCSIGN

Wenn für ein Terminal die KDCDEF-Anweisung LTERM...,USER=username angegeben wurde, dann verhält sich openUTM nach dem Verbindungsaufbau so, als ob der Benutzer seine Berechtigung schon erfolgreich nachgewiesen hätte, d.h. er muss kein KDCSIGN eingeben. Wenn für diese Benutzerkennung die Eingabe einer Ausweiskarte oder eines dunkelgesteuerten Passworts vorgeschrieben ist, dann fordert openUTM diese Eingabe(n) vom Benutzer an.

Nach Eingabe von KDCOFF BUT kann man an diesem Terminal auch unter einer anderen Kennung arbeiten (siehe Abschnitt „Abmelden von der UTM-Anwendung“).

8.1.2 Anmeldeverfahren für UPIC-Clients und TS-Anwendungen

UPIC-Clients und TS-Anwendungen sind Clients, die mit Partnertyp UPIC-R, APPLI oder SOCKET generiert wurden, und im Falle von Partnertyp SOCKET über das UTM Socket Protokoll (USP) kommunizieren.

Bei UPIC-Clients wird die Verbindung durch den Client und bei TS-Anwendungen durch den Client oder durch openUTM aufgebaut, wobei der Verbindungsaufbau durch openUTM nur möglich ist, wenn die TS-Anwendung explizit mit einer PTERM-Anweisung generiert ist.

Falls der Client die Verbindung aufbaut, muss dieser den Namen bzw. im Falle von Partnertyp SOCKET die Portnummer der UTM-Anwendung sowie Rechnernamen und/oder Rechneradresse kennen.

Nach einem erfolgreichen Verbindungsaufbau wird ein UPIC-Client oder eine TS-Anwendung in zwei Schritten angemeldet:

1. Implizites Anmelden über eine **Verbindungs-Benutzerkennung**

   Eine Verbindungs-Benutzerkennung ist dem LTERM-Partner einer TS-Anwendung oder eines UPIC-Clients fest zugeordnet und wird bei der UTM-Generierung explizit oder implizit erzeugt:

   - Explizit durch die Angabe bei USER= in der LTERM-Anweisung.
     Für eine so definierte Verbindungs-Benutzerkennung können mit der KDCDEF-Anweisung USER weitere Eigenschaften festgelegt werden.
   - Implizit durch openUTM, wenn in der LTERM-Anweisung kein USER angegeben wurde oder wenn es sich um einen LTERM-Pool handelt (TPOOL-Anweisung). Als Name der Verbindungs-Benutzerkennung wird dann der LTERM-Name genommen; bei einem LTERM-Pool setzt sich der LTERM-Name aus dem generierten Präfix und einer laufenden Nummer zusammen, z.B. UPIC0025. Für LTERM-Pools können der Verbindungs-Benutzerkennung mit TPOOL ...USER-KSET= spezielle Keycodes zugeordnet werden. Damit lassen sich die Zugriffsmöglichkeiten der Verbindungs-Benutzerkennung einschränken.

   Folgt keine Anmeldung unter einer echten Benutzerkennung, wird die vorläufige Anmeldung der Verbindungs-Benutzerkennung zu einer endgültigen. Dies wird mit einer Meldung protokolliert. Bei UPIC-Clients wird diese Meldung auch ausgegeben, wenn sich dieser anschließend unter einer echten Benutzerkennung anmeldet.
2. Explizites Anmelden über eine **echte Benutzerkennung** (optional)

   Das Verhalten von UPIC-Clients und TS-Anwendungen ist dabei unterschiedlich:

   - Bei UPIC-Clients müssen die Benutzerkennung und die Berechtigungsdaten in den jeweiligen UPIC-Schnittstellenaufrufen gesetzt werden, UPIC übergibt diese Werte anschießend an openUTM. openUTM führt dann die Anmeldung für die übergebene Benutzerkennung durch. Diese ersetzt die Verbindungs-Benutzerkennung für die Dauer der Conversation.

     Am Ende der Conversation wird der Benutzer wieder abgemeldet, falls die Benutzerkennung mit SIGNON OMIT-UPIC-SIGNOFF=NO generiert ist. Ist die Benutzerkennung mit SIGNON OMIT-UPIC-SIGNOFF=YES generiert, dann bleibt der Benutzer nach Ende der Conversation angemeldet und wird erst abgemeldet,

     - wenn vor dem Start einer neuen UPIC Conversation über dieselbe Verbindung im UPIC-Protokoll eine andere Benutzerkennung übergeben wird,
     - oder wenn die Verbindung abgebaut wird.

     Übergibt der UPIC-Client in den UPIC-Schnittstellenaufrufen keine Berechtigungsdaten, dann ist das Anmelden über eine echte Benutzerkennung nur mit einem entsprechenden Anmelde-Vorgang möglich, siehe "Anmeldeverfahren mit Anmelde-Vorgängen".

- Über eine Transportsystemverbindung kann sich ein Benutzer unter einer echten Benutzerkennung nur dann anmelden, wenn für die Anwendung ein entsprechender Anmelde-Vorgang generiert ist, siehe "Anmeldeverfahren mit Anmelde-Vorgängen". Eine Anmeldung mit einer echten Benutzerkennung über den Standard-Anmelde-Dialog ist nicht möglich.

  Wurde eine TS-Anwendung über eine echte Benutzerkennung angemeldet, dann ersetzt diese Benutzerkennung die Verbindungs-Benutzerkennung für die gesamte Dauer der Verbindung.

Sowohl für UPIC-Clients als auch für TS-Anwendungen bleibt die Verbindungs-Benutzerkennung mindestens so lange angemeldet wie die echte Benutzerkennung. Tritt ein Verbindungsverlust auf, kann das dazu führen, dass ein erneuter Verbindungsaufbau abgelehnt wird, wenn unter der echten Benutzerkennung noch ein Programm abläuft und deshalb auch die Verbindungs-Benutzerkennung als angemeldet gilt. In diesem Fall muss der Benutzer mit einer Neuanmeldung warten, bis das Programm beendet ist.

8.1.3 Anmeldeverfahren für OSI TP-Partner

Damit sich ein OSI TP-Partner an die UTM-Anwendung anmelden kann, muss der Partner die Adresse des OSI TP-Zugriffspunktes der UTM-Anwendung kennen. Diese Daten werden im OSI TP-Partner konfiguriert.

Bei OSI TP-Partnern kann die Initiative zum Verbindungsaufbau sowohl vom Partner als auch von openUTM kommen. Dabei können über eine logische Verbindung mehrere parallele Verbindungen aufgebaut werden, die Associations genannt werden. Jeder Association ist ein Association-Name zugeordnet.

Nach erfolgreichem Verbindungsaufbau wird der Client zunächst unter seinem Association-Namen angemeldet. Dieser wird zusammengesetzt aus dem in OSI-LPAP...,ASSOCIATION-NAMES= angegebenen Namen sowie einer laufenden Nummer, z.B. ASSOC03.

Wenn ein Application Context, der die abstrakte Syntax UTMSEC enthält, für die OSI TP-Kommunikation zwischen beiden Partnern generiert ist (bei openUTM mit dem Parameter APPLICATION-CONTEXT der OSI-LPAP-Anweisung), kann der Kommunikationspartner in den jeweiligen Protokollfeldern eine echte Benutzerkennung und Berechtigungsdaten übergeben. openUTM führt dann die Anmeldung für die übergebene Benutzerkennung durch. Diese Anmeldung gilt für die Dauer des OSI TP-Dialogs. Am Ende des OSI TP-Dialogs wird der Benutzer wieder abgemeldet.

Wird keine echte Benutzerkennung übergeben, dann bleibt der Client unter seinem Association-Namen angemeldet.

8.1.4 Anmeldeverfahren für HTTP-Clients

HTTP-Clients sind Clients, die mit PTYPE=SOCKET generiert wurden, und über das HTTP-Protokoll kommunizieren.

Damit sich ein HTTP-Client an die UTM-Anwendung anmelden kann, muss er die Adresse des Transportsystem-Endpunkts der UTM-Anwendung kennen. Bei HTTP-Clients geht die Initiative zum Verbindungsaufbau immer vom Client aus, dabei sendet ein HTTP-Client einen HTTP-Request an den Transportsystem-Endpunkt.

Nach einem erfolgreichen Verbindungsaufbau wird ein HTTP-Client in zwei Schritten angemeldet:

1. Implizites Anmelden über eine **Verbindungs-Benutzerkennung**

   Eine Verbindungs-Benutzerkennung ist dem LTERM-Partner eines HTTP-Clients fest zugeordnet und wird bei der UTM-Generierung explizit oder implizit erzeugt:

   - Explizit durch die Angabe bei USER= in der LTERM-Anweisung.
     Für eine so definierte Verbindungs-Benutzerkennung können mit der KDCDEF-Anweisung USER weitere Eigenschaften festgelegt werden.
   - Implizit durch openUTM, wenn in der LTERM-Anweisung kein USER angegeben wurde oder wenn es sich um einen LTERM-Pool handelt (TPOOL-Anweisung). Als Name der Verbindungs-Benutzerkennung wird dann der LTERM-Name genommen; bei einem LTERM-Pool setzt sich der LTERM-Name aus dem generierten Präfix und einer laufenden Nummer zusammen, z.B. HTTP0025. Für LTERM-Pools können der Verbindungs-Benutzerkennung mit TPOOL ...USER-KSET= spezielle Keycodes zugeordnet werden. Damit lassen sich die Zugriffsmöglichkeiten der Verbindungs-Benutzerkennung einschränken.

   Folgt keine Anmeldung unter einer echten Benutzerkennung, wird die vorläufige Anmeldung der Verbindungs-Benutzerkennung zu einer endgültigen. Dies wird mit einer Meldung protokolliert.

2. Explizites Anmelden über eine **echte Benutzerkennung** (optional)

   Ein HTTP-Client kann in einem HTTP-Request im Header-Feld Authorization Authentifizierungsdaten an die UTM-Anwendung senden. Ist dieser Header angegeben, dann muss der Wert mit “basic “ beginnen und <userid>:<password> muss base64 codiert sein. Andernfalls wird der Request mit Status-Code "400 invalid http header format“ zurückgewiesen.
   openUTM prüft die übergebenen Authentifizierungsdaten. Schlägt die Prüfung fehl, dann lehnt openUTM die Anmeldung mit "401 user not authorized" ab. Ansonsten wird der Benutzer angemeldet und kann die Services der Anwendung aufrufen, siehe "Vorgänge vom HTTP-Client aus starten".
   Die UTM-Anwendung kann auch verlangen, dass der HTTP-Client Authentifizierungsdaten sendet. Dies kann entweder für alle HTTP-Requests, die über einen Transportsystem-Endpunkt empfangen werden, konfiguriert werden, indem der Parameter USER-AUTH im KDCDEF-Statement BCAMAPPL auf den Wert *BASIC gesetzt oder für die Path-Angabe eines HTTP-Requests, indem der Parameter USER-AUTH im KDCDEF-Statement HTTP-DESCRIPTOR auf den Wert *BASIC gesetzt wird. In diesem Fall fordert openUTM die Authentifizierungsdaten vom Client an, wenn der Client keine Authentifizierungsdaten mitgesendet hat.

8.1.5 Anmeldeverfahren im Internet über Web Services (WS4UTM)

Ein Service einer UTM-Anwendung kann über WS4UTM von Internet Service-Clients aus aufgerufen werden. Damit kann ein Benutzer über das Web auf bestimmte Services einer UTM-Anwendung zugreifen.

Die Anmeldung über WS4UTM kann durch den Web Service-Client gestaltet werden:

1. Der Benutzer gibt in seinem Web Service-Client einen WebService-Namen und eine Methode an. Durch die Konfiguration ist der Web Service fest mit einer UTM-Anwendung verknüpft. Die Verbindung zur UTM-Anwendung wird über UPIC aufgebaut. Eventuell führt der Web Service-Client zuvor einen eigenen Zwischendialog, z.B. für einen Berechtigungsnachweis.
2. Der Benutzer muss eventuell wie beim Terminal die UTM-Benutzerkennung und ggf. das Passwort angeben. Ob der Benutzer einen derartigen Berechtigungsdialog führen muss und wie dieser aussieht, hängt jedoch von der Gestaltung des Web Service-Clients ab. Es ist z.B. möglich, die UTM-Benutzerkennung/-Passwort im Web Service-Client zu „verstecken“ oder sie in der Konfiguration des Web Services vorzugeben, so dass der Berechtigungsdialog intern abläuft.
3. Die Auftragsdaten (TAC und Benutzerdaten) werden zusammen mit den Berechtigungsdaten über http/Soap an einen Web Service-Server und dann über die UPIC-Verbindung an die UTM-Anwendung gesendet. Nach Rückgabe der Antwort an den Web Service-Client wird die UPIC-Verbindung wieder abgebaut.

Als Web Service-Server wird der Axis-Server von Apache verwendet.

Die Kommunikation erfolgt mit Soap-Nachrichten und http-Protokoll über Apache Tomcat und Axis. Für den Anschluss an die UTM-Anwendung nutzt WS4UTM die UPIC-Schnittstelle von openUTM.

 Nähere Informationen finden Sie im Handbuch „WebServices for openUTM“.

8.1.6 Anmeldeverfahren im World Wide Web über WebTransactions

Ein Client kann sich über WebTransactions aus dem Internet an eine UTM-Anwendung angeschließen. Damit kann ein Benutzer über einen Browser auf die Services einer UTM-Anwendung zugreifen.

Die Anmeldung kann durch die WebTransactions-Anwendung gestaltet werden:

1. Der Benutzer gibt in seinem Browser die URL der WebTransactions-Anwendung an. Danach wird die Verbindung zur UTM-Anwendung aufgebaut. Eventuell führt die Web-Transactions-Anwendung zuvor einen eigenen Zwischendialog, z.B. für einen Berechtigungsnachweis für den Zugang zur WebTransactions-Anwendung.
2. Der Benutzer muss eventuell wie beim Terminal die UTM-Benutzerkennung und ggf. das Passwort angeben. Ob der Benutzer einen derartigen Berechtigungsdialog führen muss und wie dieser aussieht, hängt jedoch von der Gestaltung der WebTransactions-Anwendung ab. Es ist z.B. möglich, die UTM-Benutzerkennung/-Passwort in der WebTransactions-Anwendung zu „verstecken“, so dass der Berechtigungsdialog intern abläuft und der Benutzer nach der Eingabe der URL sofort angemeldet ist.

Anschließend kann der Benutzer die Services der Anwendung aufrufen, siehe "UTM-Services aufrufen".

WebTransactions nutzt für den Anschluss an die UTM-Anwendung entweder die Terminal-Schnittstelle oder die UPIC-Schnittstelle von UTM.

Nähere Informationen finden Sie in den WebTransactions-Handbüchern „Anschluss an openUTM-Anwendungen über UPIC“ und „Anschluss an OSD-Anwendungen“, falls die Terminal-Schnittstelle verwendet wird.

8.1.7 Mehrfach-Anmeldungen unter einer Benutzerkennung

Ist die Benutzerkennung bei der KDCDEF-Generierung mit RESTART=NO und die UTM-Anwendung mit dem Standardwert MULTI-SIGNON=YES generiert worden, dann kann ein Benutzer über verschiedene Verbindungen mehrfach bei der UTM-Anwendung angemeldet sein, allerdings nur einmal über eine Verbindung zu einem Terminal. Mehrfach-Anmeldungen sind nur für echte Benutzerkennungen möglich, **nicht** jedoch für Verbindungs-Benutzerkennungen. Näheres zu Verbindungs-Benutzerkennungen finden Sie auf "Anmeldeverfahren für UPIC-Clients und TS-Anwendungen".

Meldet sich ein Benutzer unter einer mit RESTART=YES generierten Benutzerkennung über einen HTTP-Client oder einen OSI TP-Partner an, für dessen Conversation die Functional Unit „Commit“ ausgewählt ist, ist unter dieser Benutzerkennung ebenfalls eine weitere Anmeldung möglich, weil openUTM in diesem Falle keinen Vorgangswiederanlauf durchführt und die Benutzerkennung dann so behandelt, als sei kein Wiederanlauf generiert. Das gleiche gilt, wenn sich der Benutzer über einen OSI TP-Partner anmeldet und einen asynchronen Auftrag ausführt.

Ansonsten kann sich ein Benutzer unter einer mit RESTART=YES generierten Benutzerkennung zu einem Zeitpunkt nur einmal anmelden, da die für den Vorgangswiederanlauf benötigten Ressourcen der Benutzerkennung zugeordnet werden.

Verhindern von Mehrfach-Anmeldungen für Benutzerkennungen mit RESTART=NO

Über den Parameter MULTI-SIGNON der SIGNON-Anweisung können Sie bei der UTM-Generierung festlegen, dass unabhängig von der Wiederanlaufeigenschaft ein Benutzer zu jedem Zeitpunkt nur einmal bei openUTM angemeldet sein darf.
Diese Festlegung gilt jedoch nicht für Anmeldungen über einen OSI TP-Partner zur Ausführung von Asynchron-Aufträgen.

8.1.8 Anmeldeverfahren mit Anmelde-Vorgängen

Anmelde-Vorgänge, auch als Event-Services SIGNON bezeichnet, sind selbst programmierte Vorgänge, mit deren Hilfe eigene Anmeldeverfahren definiert werden können. Anmelde-Vorgänge können von Terminals, UPIC-Clients und TS-Anwendungen genutzt werden, d.h. von Clients, die über PTERM- oder TPOOL-Anweisung generiert sind (nicht aber von HTTP-Clients).

Aufruf von Anmelde-Vorgängen

Ein Anmelde-Vorgang ist an den Anwendungsnamen gebunden. Meldet sich ein Client unter einem bestimmten Anwendungsnamen an, dann wird der zu diesem Anwendungsnamen gehörige Anmelde-Vorgang gestartet und ersetzt das in den vorigen Abschnitten geschilderten Standard-Anmeldeverfahren. Falls mit BCAMAPPL-Anweisungen mehrere Anwendungsnamen generiert sind, dann können mehrere unterschiedliche Anmelde-Vorgänge in einer Anwendung existieren. Damit lassen sich Client-spezifische Anmelde-Vorgänge erstellen, z.B. einer für Terminals, einer für UPIC-Clients und einer für TS-Anwendungen. Weitere Details finden Sie in Abschnitten „Anmelde-Vorgänge für Terminals“ bis „Anmelde-Vorgang für UPIC-Clients“.

Ist für einen Anwendungsnamen kein Anmelde-Vorgang generiert, dann durchläuft der Client das Standard-Anmeldeverfahren.

Generieren von Anmelde-Vorgängen

Anmelde-Vorgänge werden wie folgt generiert, siehe auch openUTM-Handbuch „Anwendungen generieren“:

- Mit TAC KDCSGNTC wird der Anmelde-Vorgang für den Standard-Anwendungsnamen (definiert in MAX APPLINAME) generiert.
- Mit BCAMAPPL *appliname2...,*SIGNON=*signon-tac* wird der Anmelde-Vorgang für den Anwendungsnamen *appliname2* generiert. *signon-tac* muss in einer TAC-Anweisung definiert sein.
- Sollen Anmelde-Vorgänge auch von UPIC-Clients genutzt werden können, dann muss zusätzlich SIGNON ..., UPIC=YES generiert werden.

Für jeden dieser TACs ist auch eine PROGRAM-Anweisung nötig. Dort wird der Name des Teilprogramms angegeben, das im Anmelde-Vorgang als erstes durchlaufen wird.

Programmieren von Anmelde-Vorgängen

Für die Programmierung eines Anmelde-Vorgangs gibt es die speziellen KDCS-Aufrufe SIGN ST, SIGN ON und PEND PS. Wie ein Anmelde-Vorgang programmiert werden kann und welche Regeln dabei zu beachten sind, ist ausführlich im entsprechenden Abschnitt im openUTM-Handbuch „Anwendungen programmieren mit KDCS“ beschrieben.

8.1.8.1 Anmelde-Vorgänge für Terminals

Ein Anmelde-Vorgang für Terminals setzt sich im Allgemeinen aus zwei Teilen zusammen:

Anmelde-Vorgang	
Teil 1	Teil 2
Der Vorgang fordert den Benutzer auf, sich zu legitimieren, liest mit MGET die Berechtigungsdaten und übergibt diese zur Prüfung an openUTM. Der Vorgang ist noch **keiner** Benutzerkennung zugeordnet.	openUTM hat die Berechtigungsdaten akzeptiert und hat den Anmelde-Vorgang der ermittelten Benutzerkennung zugeordnet.

Zwischen dem ersten und zweiten Teil des Anmelde-Vorgangs kann openUTM ggf. einen Zwischendialog einfügen, der über openUTM-Meldungen gesteuert wird. Dieser Zwischendialog dient dazu, eine Magnetstreifenkarte einzulesen, oder ein mit USER...,PASS=(...,DARK) generiertes Passwort. Auch wenn die Gültigkeitsdauer des Passwortes bereits abgelaufen und die Anwendung mit Grace-Sign-On generiert ist. In diesen Fällen fordert openUTM den Benutzer wie beim Standard-Anmelde-Dialog auf, diese Information einzugeben. Dieser Zwischendialog bietet dem Anwender zusätzliche Sicherheit für den Zugang zu seinen Anwendungen. Im Zwischendialog ist es möglich (wie beim Standard-Anmelde-Dialog), das bisherige Passwort durch ein neues zu ersetzen.

Spezialfälle des Anmelde-Vorgangs für Terminals

Bei der Generierung von LTERM-Partnern mit automatischem KDCSIGN und der Anmeldung über OMNIS muss der Anmelde-Vorgang entsprechend angepasst werden.

LTERM-Partner mit automatischem KDCSIGN

Der Anmelde-Vorgang erhält beim Aufruf SIGN ST die Information, dass die Benutzerkennung bereits bekannt ist. Abhängig von der UTM-Generierung kann jetzt ein Zwischendialog zum Anfordern einer Magnetstreifenkarte oder eines Passworts durchgeführt werden. Auch wenn die Gültigkeitsdauer des Passwortes bereits abgelaufen und die Anwendung mit Grace-Sign-On generiert ist.

Anmeldung über OMNIS

Eine Verbindung von OMNIS zu UTM kann mit oder ohne Multiplex Verbindung generiert werden.

Ohne Multiplex-Verbindung wird ein vorhandener Anmelde-Vorgang durchgeführt wie für eine Terminal-Verbindung.

Mit Multiplex-Verbindung kann OMNIS die Berechtigungsdaten über das PUTMMUX-Protokoll bereits mitgeben. In diesem Fall prüft UTM die Daten zuerst und startet den Anmelde-Vorgang nur dann, wenn die Daten korrekt sidn. Der Anmelde-Vorgang erhält mit dem Aufruf SIGN ST die entsprechenden Informationen. Sind die Berechtigungsdaten nicht korrekt, wird der Anmelde-Vorgang nicht gestartet, OMNIS erhält einen entsdprechenden Returncode und muss seinerseits den Benutzer am Terminal informieren.

8.1.8.2 Anmelde-Vorgang für TS-Anwendungen

Der Anmelde-Vorgang wird unter der Verbindungs-Benutzerkennung gestartet.Beim Start des Anmelde-Vorgangs ist der Benutzer unter der Verbindungs-Benutzerkennung vorläufig angemeldet.

Im Anmelde-Vorgang können über den Aufruf SIGN ON die Berechtigungsdaten einer echten Benutzerkennung übergeben werden. Wenn openUTM die Daten akzeptiert, wird der Benutzer beim ordnungsgemäßen Abschluss des Anmelde-Vorgangs unter der angegebenen Benutzerkennung angemeldet. Die Anmeldung wird abgelehnt, wenn die Berechtigungsdaten der TS-Anwendung nicht korrekt sind oder wenn unter der Verbindungs-Benutzerkennung ein Vorgang offen ist.

Ist die Anmeldung unter einer echten Benutzerkennung einmal fehlgeschlagen, dann muss in demselben Anmelde-Vorgang eine erfolgreiche Anmeldung unter einer echten Benutzerkennung folgen, ansonsten wird bei Beendigung des Anmelde-Vorgangs die Verbindung abgebaut. D.h. die Verbindungs-Benutzerkennung ist keine Rückfallstufe für einen erfolglosen Anmeldeversuch.

Wird im Anmelde-Vorgang keine Benutzerkennung übergeben, dann wird der Benutzer beim ordnungsgemäßen Abschluss des Anmelde-Vorgangs endgültig unter der Verbindungs-Benutzerkennung angemeldet.

8.1.8.3 Anmelde-Vorgang für UPIC-Clients

Beim Anmelden über einen Anmelde-Vorgang sind zwei Fälle zu unterscheiden:

- Der UPIC-Client übergibt im UPIC-Protokoll Berechtigungsdaten an openUTM. Wenn openUTM die Daten akzeptiert, dann wird der Anmelde-Vorgang unter der übergebenen echten Benutzerkennung gestartet und der Client wird beim ordnungsgemäßen Abschluss des Anmelde-Vorgangs unter dieser Benutzerkennung angemeldet.
- Falls UPIC-Client im UPIC-Protokoll keine Berechtigungsdaten übergibt, dann wird der Anmelde-Vorgang unter der Verbindungs-Benutzerkennung gestartet. Im Anmelde-Vorgang können die Berechtigungsdaten einer echten Benutzerkennung übergeben werden. Wenn openUTM diese Daten akzeptiert, wird der Benutzer beim ordnungsgemäßen Abschluss des Anmelde-Vorgangs unter dieser Benutzerkennung angemeldet. Werden keine Berechtigungsdaten übergeben, läuft die Conversation unter der Verbindungs-Benutzerkennung.

Ist die Anmeldung unter einer echten Benutzerkennung fehlgeschlagen, dann muss eine erfolgreiche Anmeldung unter einer echten Benutzerkennung folgen, ansonsten wird bei Beendigung des Anmelde-Vorgangs die Conversation beendet. D.h. die Verbindungs-Benutzerkennung ist keine Rückfallstufe für einen erfolglosen Anmeldeversuch.

Damit Client-Programme unabhängig davon eingesetzt werden können, ob die UTM-Anwendung einen Anmelde-Vorgang verwendet oder nicht, können Nachrichten vom Client, die im Anmeldevorgang nicht gelesen wurden, nach Beendigung eines Teilprogramms des Anmelde-Vorgangs mit PEND PA, PEND PR, PEND PS oder PEND FC ohne vorhergehenden MPUT im Folgeteilprogramm gelesen werden.

8.1.8.4 Anwendungsmöglichkeiten für Anmelde-Vorgänge

Anmelde-Vorgänge bieten dem Anwender eine Reihe praktischer Nutzungsmöglichkeiten, die im Folgenden skizziert werden:

- Das Anmeldeverfahren für Terminals kann als formatierter Dialog ablaufen.
- Startformate können aktuelle Daten enthalten, wie Datum, Uhrzeit, Bulletin etc.
- In einer Anwendung mit mehrsprachigen Benutzern kann man LTERM-spezifische Startformate verwenden, um einen Begrüßungsbildschirm in der jeweiligen Landessprache auszugeben. Diese Möglichkeit gibt es nur bei Terminals.
- Benutzer-spezifische Startformate sind zweckmäßig, um Informationen zu zeigen, die spezifisch für eine Benutzergruppe sind (Bulletin, Menü etc.).
- TS-Anwendungen können sich über einen Anmelde-Vorgang mit einer echten Benutzerkennung an eine UTM-Anwendung anmelden. Dadurch werden sie in das Zugangs- und Zugriffskonzept von openUTM eingebunden.
- Ein vom Benutzer eingegebener realer Name kann in eine Benutzerkennung umgesetzt werden, die per UTM-Generierung definiert ist (USER *username*).
- Im Falle eines DB/DC-übergreifenden Berechtigungskonzeptes kann man im 2. Teil des Anmelde-Vorgangs mit einem Datenbankaufruf das aktuelle Berechtigungsprofil für diesen USER aus der Datenbank holen und evtl. in einem Benutzer-spezifischen Langzeitspeicher (ULS) speichern.
- Der Anmelde-Vorgang kann den Benutzer im 2. Teil zur Änderung seines Passwortes auffordern, z.B. weil die Zeitspanne überwacht wird, in welcher der Benutzer dasselbe Passwort verwenden darf.
- Es kann eine Statistik über alle versuchten und erfolgreichen Anmeldungen erstellt werden.
- Das Formatsteuerungssystem FHS bietet die Möglichkeit, für Terminals die P-Tasten über ein Globalattribut eines #-Formats zu laden. Der Anmelde-Vorgang kann im 2. Teil die P-Tasten beispielsweise Benutzer-spezifisch laden.
- Auch im Falle eines nachfolgenden Vorgangswiederanlaufs kann der Anmelde-Vorgang dem Benutzer nützliche Informationen geben. Hierzu zählen Bulletin, Anzeige der Tastaturbelegung oder Anzeige des Vorgangswiederanlaufs. Dies erfordert einen zusätzlichen Dialogschritt.
- Wenn openUTM nach einem Aufruf SIGN OB (= KDCOFF BUT per Programm) den Anmelde-Vorgang startet, kann es sinnvoll sein, dass mit MGET die letzte Eingabe vom Terminal gelesen wird, wenn dort bereits neue Berechtigungsdaten eingetragen wurden.

8.1.8.5 Eigenschaften von Anmelde-Vorgängen

Ausgabe der letzten Dialog-Nachricht durch den Anmelde-Vorgang

Liegt kein Vorgangswiederanlauf vor und wird der Anmelde-Vorgang mit MPUT PM und PEND FI beendet, dann wird die letzte Dialog-Nachricht der letzten Sitzung des Benutzers ausgegeben, sofern dieser mit RESTART=YES generiert ist. Der Benutzer kann dann nach Abschluss des Anmelde-Vorgangs mit dem gleichen Bildschirm weiterarbeiten, mit dem die letzte Sitzung beendet wurde - unabhängig davon, ob dies innerhalb oder außerhalb eines Vorgangs geschah.

Meldungen

Verwendet eine UTM-Anwendung einen Anmelde-Vorgang, werden die folgenden Meldungen nicht erzeugt (und demzufolge auch nicht an SYSLOG und MSGTAC ausgegeben):

K001, K002, K004 bis K008.

Die Meldung K033 (erfolgreiche Anmeldung) wird auch bei Verwendung eines Anmelde-Vorgangs erzeugt.

Fehlversuche beim Anmelde-Vorgang

Beim Anmelde-Vorgang können Fehlversuche des Benutzers beim Anmelden abgefangen werden: Akzeptiert openUTM die eingegebenen Berechtigungsdaten des Benutzers **nicht**, dann kann der Anmelde-Vorgang den Benutzer auffordern, die Eingabe zu wiederholen. Die maximale Anzahl der Eingabeversuche ist programmierbar. Wird sie überschritten, dann sollte sich der Anmelde-Vorgang beenden. Bei TS-Anwendungen und bei Terminals baut dann UTM die Verbindung ab, bei UPIC wird nur die Conversation beendet.

openUTM zählt außerdem alle Fehlversuche von einem Client oder unter einer Benutzerkennung mit, die in ununterbrochener Folge auftreten, auch über eine Folge von Anmelde-Vorgängen hinweg. Nach einer bei der UTM-Generierung festzulegenden Anzahl von fehlerhaften Anmeldeversuchen (siehe KDCDEF-Anweisung SIGNON, Operand SILENT-ALARM im openUTM-Handbuch „Anwendungen generieren“) meldet openUTM dieses Ereignis nach SYSLOG (stiller Alarm, Meldung K094). Anmeldeversuche unberechtigter Personen können aufgedeckt und mit einer MSGTAC-Routine abgewehrt werden.

Fehlverhalten des Anmelde-Vorgangs

openUTM kontrolliert, ob die Regeln für den Anmelde-Vorgang eingehalten werden. Das bietet auch einen Schutz gegen eventuelle Manipulationen der Teilprogramme des Anmelde-Vorgangs. Bei Fehlern dieser Art bricht openUTM den Anmelde-Vorgang mit PEND ER ab. Danach wird bei TS-Anwendungen und Terminals die Verbindung abgebaut; bei UPIC wird nur die Conversation beendet.

8.1.8.6 Beispielprogramme für Anmelde-Vorgang

Zusammen mit openUTM werden Teilprogramme als COBOL-Quellprogramme ausgeliefert, die einen fertigen Anmelde-Vorgang mit formatierter Schnittstelle zum Terminal realisieren. Dieser Anmelde-Vorgang ist für alle Generierungsvarianten geeignet. Das verwendete Format enthält englische Texte.

Der Anwender kann diese Vorlage nach seinen Wünschen abändern und erhält so auf einfache Weise ein Anmeldeverfahren mit formatierter Schnittstelle zum Benutzer. Er muss dadurch mit der Programmierung nicht völlig neu beginnen.

8.1.9 Verhalten bei gesperrten Clients/LTERM-Partnern

Verhalten bei gesperrten Clients

Clients können per UTM-Generierung (PTERM...,STATUS=OFF) oder per Administration gesperrt werden. Das Sperren eines Client hat folgende Wirkungen:

- Ein Verbindungswunsch wird abgelehnt.
- Eine bestehende Verbindung bleibt erhalten; die Sperre wird erst dann wirksam, wenn von diesem Client ein neuer Verbindungswunsch kommt.

Verhalten bei gesperrten LTERM-Partnern

LTERM-Partner können per UTM-Generierung (LTERM...,STATUS=OFF) oder per Administration gesperrt werden.

Bei UPIC-Clients und TS-Anwendungen wirkt das Sperren des LTERM-Partners wie das Sperren des Clients.

Bei Terminals hat das Sperren eines LTERM-Partners folgende Wirkungen:

- Der Verbindungswunsch wird ausgeführt, aber nach dem Verbindungsaufbau wird die Meldung ausgegeben:

  ```
  K027 LTERM-Partner <ltermname> gesperrt - Administrator verstaendigen oder KDCOFF eingeben.
  ```

- Eine bestehende Verbindung bleibt erhalten, die nächste Eingabe vom Terminal wird mit der Meldung K027 quittiert.

8.2 Anmeldeverfahren ohne Benutzerkennungen

Bei UTM-Anwendungen, für die keine Benutzerkennungen generiert sind, nimmt openUTM keine Berechtigungsprüfung vor. Die Clients werden unter ihrem LTERM-Namen bzw. Associations-Namen angemeldet. UPIC-Clients, HTTP-Clients und OpenCPIC-Clients dürfen in diesem Fall keine Benutzerkennung übergeben.

Verwendet die UTM-Anwendung Anmelde-Vorgänge ("Anmeldeverfahren mit Anmelde-Vorgängen"), dann kann damit eine anwendungseigene Berechtigungsprüfung durchgeführt werden, zum Beispiel anhand einer Datenbank mit Berechtigungsdaten.

Wird kein Anmelde-Vorgang verwendet, dann kann der Benutzer nach erfolgreichem Verbindungsaufbau zu der UTM-Anwendung sofort mit dieser Anwendung arbeiten. Bei Terminals und TS-Anwendungen (Clients mit Partnertyp APPLI oder SOCKET, die im Falle von Partnertyp SOCKET über das UTM Socket Protokoll (USP) kommunizieren) erhält der Benutzer von openUTM eine Ausgabe, die davon abhängt, ob für diesen LTERM-Partner noch ein offener Vorgang bekannt ist:

- Wenn für den LTERM-Partner in der Anwendung kein offener Vorgang bekannt ist, gibt openUTM folgende Meldung aus:

  ```
  K001 Verbunden mit Anwendung <anwendungsname> - Bitte Eingabe
  ```

  Bei Terminals wird, falls generiert, das Startformat zu diesem LTERM-Partner ausgegeben. Der Benutzer kann dann Vorgänge starten und UTM-Benutzer-Kommandos eingeben.
- Wenn für diesen LTERM-Partner in der Anwendung noch ein offener Vorgang bekannt ist, so bekommt der Benutzer die Ausgabe vom letzten Sicherungspunkt des unterbrochenen Vorgangs auf den Bildschirm, und er kann den Vorgang fortsetzen. Sehen Sie dazu auch openUTM-Handbuch „Anwendungen programmieren mit KDCS“. Dafür ist unter anderem Voraussetzung, dass für diesen LTERM-Partner RESTART=YES generiert wurde. Das heißt aber auch, dass der Benutzer eventuell den Vorgang eines anderen Benutzers fortsetzen kann.

Beachten Sie, dass openUTM in einer Anwendung ohne Benutzerkennungen einen Vorgang an den LTERM-Partner koppelt. Ein unterbrochener Vorgang kann deshalb auch nur vom selben Client fortgesetzt werden. Es sei denn, die Zuordnung LTERM-Partner und physikalischer Client (festgelegt in der PTERM-Anweisung) wird per Administration, z.B. mit dem Administrationskommando KDCSWTCH, geändert.

Das Verhalten bei gesperrten Clients ist das gleiche wie mit Benutzerkennungen, siehe "Verhalten bei gesperrten Clients/LTERM-Partnern".

> **ACHTUNG!**
>
> In einer UTM-Anwendung ohne Benutzerkennungen hat jeder Benutzer die Administrationsberechtigung.

8.3 UTM-Services aufrufen

Ist die UTM-Berechtigungsprüfung erfolgreich verlaufen, so ist der Benutzer berechtigt, mit der UTM-Anwendung zu arbeiten, d.h er kann neue Vorgänge (=Services) starten (siehe unten) oder offene Vorgänge fortsetzen.

Die Abschnitte „Vorgänge vom Terminal aus starten“ bis „Vorgänge von TS-Anwendungen aus starten“ zeigen, wie bei den einzelnen Client-Typen neue Vorgänge gestartet werden. Der Abschnitt „Vorgangswiederanlauf“ beschreibt die Situation, wenn für diese Benutzerkennung in der Anwendung noch ein offener Vorgang bekannt ist.

8.3.1 Vorgänge vom Terminal aus starten

Nach erfolgreicher Anmeldung kann der Benutzer einen Vorgang starten, indem er einen Transaktionscode (TAC) eingibt oder eine entsprechend generierte Funktionstaste drückt.

Starten eines Vorgangs durch Eingabe eines Transaktionscodes

Wenn kein Anmelde-Vorgang durchlaufen wird, kann der Benutzer folgende Situationen vorfinden:

- Im Zeilenmodus gibt openUTM die folgende Meldung aus:

  ```
  K008 Anmeldung akzeptiert - Bitte Eingabe
  ```

  Der Benutzer kann einen Vorgang starten, indem er einen TAC und eventuell eine Nachricht eingibt. Die ersten acht Zeichen seiner Eingabe werden von openUTM als TAC interpretiert. Ist der TAC kürzer als 8 Zeichen, dann muss er durch ein Leerzeichen von der Nachricht getrennt sein.
- Wenn für den Benutzer ein Startformat generiert wurde, dann muss er die entsprechenden Felder des Startformats ausfüllen. Das Feld mit dem TAC, auch Steuerfeld genannt, steht dabei nicht notwendigerweise an erster Stelle.

Wenn ein Anmelde-Vorgang durchlaufen wird, dann bestimmt der Anmelde-Vorgang den nächsten Schritt. Der Benutzer erhält dann eine Ausgabe im Format- oder Zeilenmodus oder es wird direkt ein Vorgang gestartet.

Starten eines Vorgangs über eine Funktionstaste

Ein TAC kann per UTM-Generierung auf eine Funktionstaste gelegt werden, z.B. F10, siehe KDCDEF-Anweisung SFUNC. Wenn der Benutzer diese Taste betätigt, dann wird der Vorgang gestartet, unabhängig davon, ob ein Benutzer im Zeilenmodus arbeitet oder ein Startformat vor sich hat.

Eingabe ungültiger Transaktionscodes

Wenn der Benutzer einen falschen TAC eingibt, dann erhält er die Meldung

```
K009 Der Transactionscode <tac> ist ungültig - Bitte Eingabe
```

Falls in der Anwendung ein Dialog-Vorgang BADTACS generiert ist, dann wird stattdessen der Vorgang BADTACS gestartet. Nachdem der Dialog-Vorgang BADTACS beendet wurde, bleibt der Benutzer angemeldet und kann wie oben beschrieben einen Vorgang starten.

8.3.2 Vorgänge vom UPIC-Client und OSI TP-Partner aus starten

Nach dem Verbindungsaufbau können die OSI TP-Partner oder UPIC-Clients Vorgänge starten. Dazu wird der TAC durch den Client gesetzt, z.B. über die Funktion *Set_TP_Name* der CPI-C-Schnittstelle oder durch einen entsprechenden Eintrag in der Side Information Datei. Dieser TAC wird an openUTM übergeben, eventuell zusammen mit Berechtigungsdaten. Nach erfolgreicher Berechtigungsprüfung gilt:

- Bei OSI TP-Partner und bei UPIC-Clients ohne Anmelde-Vorgang wird der zum TAC gehörige Vorgang sofort gestartet.
- Bei UPIC-Clients mit Anmelde-Vorgang wird der zum TAC gehörige Vorgang erst nach dem Ende des Anmelde-Vorgangs gestartet.

  Am Ende des Vorgangs wird der Benutzer wieder abgemeldet, wenn er sich für diesen Vorgang unter einer echten Benutzerkennung angemeldet hat.
  Dies gilt nicht für UPIC-Clients in einer UTM-Anwendung, die mit
  SIGNON OMIT-UPIC-SIGNOFF=YES generiert wurde, siehe "Anmeldeverfahren für UPIC-Clients und TS-Anwendungen".

8.3.3 Vorgänge vom HTTP-Client aus starten

Nach dem Verbindungsaufbau können HTTP-Clients Vorgänge starten. Details zur Angabe der Berechtigungsdaten und des TACs sind im openUTM-Handbuch "Konzepte und Funktionen" sowie im openUTM-Handbuch "Anwendungen programmieren mit KDCS" beschrieben.

8.3.4 Vorgänge von TS-Anwendungen aus starten

TS-Anwendungen verhalten sich ähnlich wie Terminals:

- Ohne Anmelde-Vorgang erhält die TS-Anwendung die Meldung K001, falls dieser Meldung das Meldungsziel PARTNER zugewiesen wurde, siehe Beschreibung des Tools KDCMMOD im openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“.

  Anschließend kann die TS-Anwendung einen Vorgang starten, indem sie einen TAC und eventuell eine Nachricht an die UTM-Anwendung übergibt. Dazu werden die ersten 8 Zeichen der Nachricht als TAC interpretiert. Ist der TAC kürzer als 8 Zeichen, muss er durch Leerzeichen von der Nachricht getrennt sein.
- Wird ein Anmelde-Vorgang durchlaufen, dann bestimmt dieser den nächsten Schritt. Der Anmelde-Vorgang kann entweder direkt einen Vorgang starten oder eine Ausgabenachricht an die TS-Anwendung schicken. Wird eine Ausgabenachricht an die TS-Anwendung geschickt, dann muss die nächste Nachricht in den ersten 8 Zeichen einen TAC enthalten, d.h. es gilt das gleiche wie ohne Anmelde-Vorgang, siehe oben.

Nach dem Beenden des Vorgangs kann der nächste Vorgang gestartet werden.

8.3.5 Vorgangswiederanlauf

Ein Vorgangswiederanlauf wird in der Regel durchgeführt, wenn:

- sich ein Client unter einer Benutzerkennung anmeldet, die mit RESTART=YES generiert ist,
- und für diese Benutzerkennung in der Anwendung noch ein offener Vorgang bekannt ist.

Wurde am letzten Sicherungspunkt eine Nachricht an den Client gesendet, so sendet openUTM diese Nachricht erneut an den Client. Der Benutzer kann den Vorgang anschließend fortsetzen. Ansonsten wird der offene Vorgang sofort fortgesetzt.

Abhängig von der Art des Client und vom Anmeldeverfahren gilt für den Vorgangswiederanlauf Folgendes:

- Standard-Anmeldeverfahren für Terminals und TS-Anwendungen:
  openUTM führt den Vorgangswiederanlauf automatisch durch.
- Standard-Anmeldeverfahren für UPIC-Clients und OSI TP-Partner:
  Der Client muss eine spezielle Conversation starten, die mittels des UTM-Benutzerkommandos KDCDISP den Wiederanlauf anfordert (siehe z.B. Handbuch „openUTM-Client für das Trägersystem UPIC“). Über OSI TP-Clients ist kein Fortsetzen eines Vorgangs möglich, wenn die Functional Unit „Commit“ ausgewählt wurde. Existiert ein offener Vorgang für den Benutzer und es wird kein Vorgangswiederanlauf angefordert, so wird der offene Vorgang abormal beendet.
- Anmelden über Anmelde-Vorgang:
  Der Anmelde-Vorgang muss den Wiederanlauf initiieren oder den offenen Vorgang abnormal beenden.

> **i** In einer Anwendung mit Benutzerkennungen ist ein Vorgang an die Benutzerkennung gebunden. Deshalb kann der Benutzer einen unterbrochenen Vorgang auch an einem anderen Client fortsetzen, sofern der LTERM-Partner des Clients die Berechtigung dazu hat und der Typ des Clients gleich bleibt.

8.4 Zugriffskontrolle von openUTM

openUTM bietet zusätzlich zum Zugangsschutz durch Benutzerkennungen ein ausgefeiltes Zugangs- und Zugriffskonzept. Damit lässt sich steuern, welcher Benutzer über welche LTERM-Partner auf welche Services der UTM-Anwendung zugreifen darf.

Sie können wählen zwischen einer benutzerorientierten Variante (**Lock-/Keycode**-Konzept) und einer rollenorientierten Variante (**Access-List-**Konzept). Diese Varianten werden mit Hilfe von Lockcodes, Access-Lists, Keysets und Keycodes generiert:

- Ein Service wird entweder mit Lockcodes (Lock-/Keycode-Konzept) oder mit einer Access-List (Access-List-Konzept) geschützt (TAC-Anweisung, LOCK= bzw. ACCESS-LIST=).
- Eine Benutzerkennung erhält ein Keyset mit einem oder mehreren Keycodes (USER-Anweisung, KSET=). Die Keycodes definieren die Berechtigungen.
- Ein LTERM-Partner erhält ein Keyset mit einem oder mehreren Keycodes sowie Lockcodes, wenn das Lock-/Keycode-Konzept verwendet wird (LTERM- bzw. TPOOL-Anweisung, Operanden KSET= und LOCK=).
- Keysets werden eigens in KSET-Anweisungen definiert

Die folgende Tabelle zeigt für die beiden Konzept-Varianten, unter welchen Voraussetzungen ein Benutzer sich anmelden kann und wann er einen Vorgang starten oder fortsetzen darf (nach einem Vorgangswiederanlauf).

Tätigkeit	**Voraussetzungen**	
	Lock-/Keycode-Konzept	**Access-List-Konzept**
Anmelden über bestimmten LTERM-Partner	Ein Keycode der Benutzerkennung stimmt mit dem Lockcode des LTERM-Partners überein.	Anmelden immer möglich.
Starten eines Vorgangs	Benutzerkennung und LTERM-Partner besitzen einen Keycode, der mit dem Lockcode des TACs übereinstimmt.	Benutzerkennung und LTERM-Partner besitzen jeweils einen Keycode, der in der Access-List des TACs enthalten ist. Die Keycodes von Benutzerkennung und LTERM müssen nicht identisch sein.
Vorgang fortsetzen (nach Vorgangswiederanlauf)	Ein Keycode des LTERM-Partners, über den der Benutzer den Vorgang fortsetzt, muss mit dem Lockcode des Folge-TACs übereinstimmen.	Ein Keycode des LTERM-Partners, über den der Benutzer den Vorgang fortsetzt, muss in der Access-List des Folge-TACs enthalten sein.

Meldungen bei fehlenden Berechtigungen

Wenn Berechtigungen fehlen, kann der Terminalbenutzer folgende Meldungen erhalten (beim Anmelde-Vorgang kommt ein entsprechender Returncode):

`K005 Die Benutzerkennung <`*user*`> ist gesperrt - Bitte KDCSIGN`

wenn der Keycode des Benutzers nicht mit dem Keycode des LTERM-Partners übereinstimmt (Anmelde-Vorgang: Returncode U02).

```
K009 Der Transaktionscode <tac> ist ungueltig - Bitte Eingabe
```

wenn der Benutzer oder das LTERM nicht die Berechtigung haben, den Vorgang zu starten. Wenn ein BADTAC-Vorgang generiert ist, dann wird stattdessen der BADTAC-Vorgang gestartet.

```
K123 LTERM hat nicht die Berechtigung den Vorgang fortzusetzen - Bitte KDCSIGN
```

wenn der LTERM-Partner, über den sich der Benutzer beim Vorgangswiederanlauf angemeldet hat, nicht die Berechtigung hat, den Folge-TAC zu starten (Anmelde-Vorgang: Returncode U16). Diese Meldung kann insbesondere dann auftreten, wenn ein Benutzer den Vorgang von einem anderen Terminal und damit einem anderen LTERM fortsetzt.

Weitere Informationen finden Sie im openUTM-Handbuch „Konzepte und Funktionen“ sowie im openUTM-Handbuch „Anwendungen generieren“.

8.5 Abmelden von der UTM-Anwendung

Die folgenden Abschnitte beschreiben die verschiedenen Möglichkeiten, wie sich ein Client bei der UTM-Anwendung abmelden kann oder ein Client durch UTM abgemeldet wird. Dabei unterscheiden sich Terminals von allen anderen Clients, denn nur von Terminals aus können sich Benutzer explizit von der Anwendung abmelden.

Abmelden bei Zeitüberschreitung - Zeitüberwachung

Bei der UTM-Generierung können maximale Wartezeiten festgelegt werden durch:

- die Operanden TERMWAIT= (PEND KP-Timer) und PGWTTIME= (PGWT-Timer) in der KDCDEF-Steueranweisung MAX
- den Operanden IDLETIME= (Transaktionsende-Timer) der PTERM- oder TPOOL-Anweisung bzw. der OSI-LPAP-Anweisung (bei OSI TP-Partnern).

Wenn eine mit diesen Timern eingestellte Wartezeit abgelaufen ist, dann wird bei Terminals folgende Meldung ausgegeben:

```
K021 Eine Eingabe ist nicht in der vorgegebenen Zeit erfolgt
```

Anschließend meldet openUTM die Benutzerkennung ab und baut die Verbindung zum Client ab. Der Client kann sich danach wieder bei der Anwendung anmelden und einen eventuell offenen Vorgang fortsetzen, siehe Abschnitt „Vorgangswiederanlauf“.

Abmelden mit KDCOFF-Kommando

Ein Terminal-Benutzer kann sich durch Eingabe des UTM-Kommandos KDCOFF bzw. KDCOFF BUT von der UTM-Anwendung abmelden. Sehen Sie hierzu auch das UTM-Benutzerkommando KDCOFF auf "KDCOFF - Abmelden von einer UTM-Anwendung".

KDCOFF aus einem Programm

openUTM bietet die Funktionsaufrufe SIGN OF und SIGN OB, mit denen in einem Dialog-Teilprogramm die Wirkung des Benutzerkommandos KDCOFF bzw. KDCOFF BUT ausgelöst werden kann. SIGN OF/OB ist für Terminals, UPIC-Clients und TS-Anwendungen möglich. In Teilprogrammen, die für einen OSI TP-Partner laufen, sind diese Aufrufe nicht erlaubt.

SIGN OF und SIGN OB wirken wie folgt:

SIGN-Aufruf	Kommando	Auswirkung
SIGN OF	KDCOFF	openUTM baut Verbindung zum Client ab.
SIGN OB	KDCOFF BUT	Für Terminals bleibt die Verbindung erhalten, der Benutzer wird abgemeldet. Bei UPIC-Clients und TS-Anwendungen wird die Verbindung abgebaut (wie SIGN OF).

Der Aufruf wirkt bei Terminals und UPIC-Clients/TS-Anwendungen unterschiedlich:

- Bei Terminals gibt openUTM zunächst die MPUT-Nachricht und die Meldung K095 an das Terminal aus. Erst durch die nächste (beliebige) Eingabe vom Terminal wird der Benutzer abgemeldet und (bei SIGN OF) die Verbindung abgebaut.

- Bei UPIC-Clients und TS-Anwendungen wird die MPUT-Nachricht gesendet und anschließend sofort der Verbindungsabbau initiiert.

Im Folgenden werden einige Anwendungsmöglichkeiten für den Funktionsaufruf SIGN OF/OB skizziert:

- Anwendungen mit besonderen Sicherheitsanforderungen. Nach dem Anmelden darf ein Benutzer nur einen einzigen Vorgang bearbeiten.
- Verbesserter Datenschutz: Der Bildschirm kann durch den letzten MPUT überschrieben werden, es bleiben keine Vorgangs-spezifischen Daten am Bildschirm stehen.
- Der Steuerteil des Bildschirms bietet als mögliche nächste Aktionen auch „Abmelden“ oder „Neu-Anmeldung“ an. Das Folgeteilprogramm erzeugt dann abhängig von der Eingabe einen Aufruf SIGN OF bzw. SIGN OB. Nach der Dialogausgabe dieses Teilprogramms und der darauf folgenden Eingabe wird entweder die Verbindung zum Terminal abgebaut oder aber der Anmelde-Vorgang gestartet.

8.6 UTM-Benutzerkommandos für Terminals

Dieser Abschnitt enthält eine Beschreibung aller UTM-Benutzerkommandos, die dem Terminal-Benutzer nach dem Anmelden zur Verfügung stehen (das zum Anmelden benötigte Kommando KDCSIGN ist auf "Standard-Anmelde-Dialog" beschrieben):

- KDCFOR, um das Basisformat auszugeben
- KDCOUT, um asynchrone Nachrichten anzufordern
- KDCDISP, um die letzte Dialog-Nachricht nochmals anzufordern
- KDCLAST, um die letzte Ausgabe zu wiederholen
- KDCOFF, um sich abzumelden

UTM-Benutzerkommandos können bei der UTM-Generierung der Anwendung mit der KDCDEF-Anweisung SFUNC einer K-/F-Taste zugeordnet werden, und können dann über Funktionstaste eingegeben werden.

8.6.1 KDCFOR - Basisformat ausgeben

Mit dem KDCFOR-Kommando kann der Benutzer ein Basisformat ausgeben. Ein Basisformat dient dazu, die Daten für einen Auftrag am Bildschirm in Formatfelder einzutragen.

Durch Eingabe von KDCFOR { *formatkennzeichen*[1] | '*formatkennzeichen*'[2] } wird das Format mit dem angegebenen Formatkennzeichen am Terminal ausgegeben. Dieses Format gilt solange als Basisformat des Benutzers, bis ein weiteres KDCFOR-Kommando mit einem anderen Formatkennzeichen eingegeben wird.

Mit dem Kommando KDCFOR ohne *formatkennzeichen* kann der Benutzer das gerade geltende Basisformat am Terminal ausgeben.

> ¡ Mit dem KDCFOR-Kommando kann man **kein** #Format ausgeben.

Im Fehlerfall gibt openUTM eine der folgenden Meldungen aus:

`K013 Fehlerhafte Eingabe im Kommando - Bitte Eingabe`

Das KDCFOR-Kommando hat nicht die vorgeschriebene Form.

`K014 Es ist kein Basisformat verfuegbar - Bitte Eingabe`

Der Benutzer hat ein KDCFOR-Kommando ohne Formatkennzeichen angegeben, es ist aber noch kein Basisformat bekannt.

`K015 Formatierungsfehler ..... - Bitte Eingabe`

Beim Versuch, das Basisformat auszugeben, trat ein Fehler auf.

`K003 Kommando KDCFOR ist in dieser Situation nicht erlaubt`

Das KDCFOR-Kommando ist auf dieser Stufe des Dialogs (mit der UTM-Anwendung) nicht erlaubt.

---

[1] wenn Präfix für *formatkennzeichen* = *

[2] wenn Präfix für *formatkennzeichen* = *, +, oder -

8.6.2 KDCOUT - Asynchrone Nachricht ausgeben

Mit dem KDCOUT-Kommando kann der Benutzer die Ausgabe asynchroner Nachrichten anfordern. Bei der UTM-Generierung wird mit dem Operanden ANNOAMSG={ YES | NO } in der LTERM bzw. TPOOL-Anweisung festgelegt, wie openUTM asynchrone Nachrichten an dieses Terminal (LTERM) bzw. diesem LTERM-Pool (TPOOL) ausgeben soll.

Ausgabe bei ANNOAMSG=Y

In diesem Fall kündigt openUTM asynchrone Nachrichten mit der folgenden Meldung an:

```
K012 nnn Nachricht(en) vorhanden
```

Die Meldung erscheint zusammen mit einer Dialogausgabe an dieses Terminal in der Systemzeile. Mit *nnn* wird die Anzahl der asynchronen Nachrichten angegeben. Der Benutzer kann diese Nachrichten mit dem Kommando KDCOUT abholen. Sind bei Eingabe von KDCOUT keine Nachrichten für das Terminal vorhanden, so meldet openUTM:

```
K020 keine Nachricht vorhanden
```

Wird eine asynchrone Nachricht mit KDCOUT abgeholt, dann wird sie durch die nächste Eingabe gelöscht, außer bei Eingabe von KDCLAST (siehe "KDCLAST - Letzte Ausgabe wiederholen").

Wenn sich der Benutzer in einem formatgesteuerten Dialog mit einem Vorgang befindet und mit KDCOUT eine asynchrone Nachricht ausgibt, so zerstört er damit das letzte Ausgabeformat. Zur Fortsetzung des Vorgangs muss er das Ausgabeformat mit dem Benutzerkommando KDCDISP (siehe "KDCDISP - Letzte Dialog-Nachricht ausgeben") wiederherstellen. Tut er das nicht, so führt openUTM selbst einen KDCDISP durch. Der Benutzer muss anschließend seine Eingabe wiederholen.

Die KDCDEF-Anweisung LTERM ..., RESTART= NO hat zur Folge, dass beim Verbindungsaufbau oder -abbau zu diesem LTERM-Partner anstehende asynchrone Nachrichten gelöscht werden.

Die Funktionsvarianten von openUTM haben folgende Auswirkungen auf die Behandlung von asynchronen Nachrichten:

- Bei UTM-S-Anwendungen bleiben asynchrone Nachrichten auch über Unterbrechungen des Anwendungslaufes hinweg gesichert, bis sie mit KDCOUT abgeholt werden.
- Bei UTM-F-Anwendungen werden asynchrone Nachrichten nur während des Anwendungslaufes gespeichert. Sie gehen mit der Beendigung des Anwendungslaufes verloren.

Ausgabe bei ANNOAMSG=N

Wenn ein Client mit einem LTERM-Partner verbunden ist, der mit LTERM...,ANNOAMSG=N oder TPOOL ..., ANNOAMSG=N generiert ist, dann ist an diesem Client das Kommando KDCOUT nicht erlaubt. ANNOAMSG=N bedeutet, dass openUTM asynchrone Nachrichten sofort, d.h. ohne vorherige Ankündigung, am Terminal ausgibt. openUTM weist deshalb das Kommando KDCOUT an dieses Terminal mit der Meldung K003 ab:

```
K003 Das Kommando KDCOUT in dieser Situation nicht erlaubt
```

In einer UTM-Anwendung mit Benutzerkennungen werden asynchrone Nachrichten frühestens nach der erfolgreich abgeschlossenen Berechtigungsprüfung ausgegeben.

Möchte der Benutzer nach Ausgabe der asynchronen Nachricht seinen Dialog mit der Anwendung fortsetzen, so muss er zuvor mit KDCDISP den Bildschirm der letzten Dialogausgabe anfordern. Andernfalls führt openUTM einen automatischen KDCDISP durch. Dieser automatische Bildschirmwiederanlauf wird von openUTM nur dann unterdrückt, wenn die asynchrone Nachricht mit demselben Format ausgegeben wurde wie die letzte Dialogausgabe.

> ¡ Bei #-Formaten erfolgt **immer** ein automatischer Wiederanlauf.

8.6.3 KDCDISP - Letzte Dialog-Nachricht ausgeben

In einer laufenden UTM-Anwendung kann sich ein Benutzer mit dem KDCDISP-Kommando die letzte Dialog-Nachricht noch einmal ausgeben lassen.

Gibt der Benutzer das KDCDISP-Kommando nach Abschluss des Anmelde-Vorgangs oder nach der Rückkehr von einem eingeschobenen Vorgang ein, dann zeigt openUTM nochmals den letzten Bildschirm der letzten Session, bzw. den letzten Bildschirm des unterbrochenen Vorgangs.

Bei Bildschirmausgaben in FHS-DE-Formaten lesen Sie bitte auch den Abschnitt „KDCDISP, KDCLAST in FHS-DE-Formaten“ (KDCLAST - Letzte Ausgabe wiederholen).

Das KDCDISP-Kommando ist in folgenden Situationen nützlich:

- Aufgrund von Übertragungsproblemen oder durch Fehlbedienung am Terminal ist der Bildschirminhalt nach einer Dialogausgabe teilweise oder vollständig zerstört.
- Der Benutzer hat während der Bearbeitung eines Vorgangs asynchrone Nachrichten am Bildschirm erhalten. Die asynchronen Nachrichten wurden entweder durch openUTM direkt gesendet oder der Benutzer hat sie mit KDCOUT selbst angefordert. Wenn der Benutzer danach den offenen Vorgang fortsetzen möchte, lässt er sich dazu mit einem KDCDISP-Kommando die letzte Dialogausgabe nochmals ausgeben.

  Wenn der Benutzer direkt in der asynchronen Formatnachricht den Vorgang fortsetzen will, erzeugt openUTM implizit einen KDCDISP, und die Eingabe ist zu wiederholen.
- Nach Beendigung und erneutem Start der UTM-Anwendung kann der Benutzer (zur Orientierung) mit dem KDCDISP-Kommando die letzte Dialogausgabe des vor Beendigung der Anwendung abgeschlossenen Vorgangs wiederholen. Das gilt aber nur, wenn es sich um eine UTM-S-Anwendung handelt und wenn der Vorgangswiederanlauf nicht explizit ausgeschaltet wurde durch die KDCDEF-Anweisung USER ..., RESTART=NO (bzw. LTERM ...,RESTART=NO, wenn die Anwendung ohne Benutzerkennungen generiert wurde).

8.6.4 KDCLAST - Letzte Ausgabe wiederholen

Das Kommando KDCLAST ermöglicht die Wiederholung der letzten Ausgabenachricht am Terminal unabhängig davon, ob es eine Dialog-Nachricht oder eine asynchrone Nachricht war. Bei Bildschirmausgaben in FHS-DE-Formaten sehen Sie auch den folgenden Abschnitt.

Wurde als letztes eine asynchrone Nachricht ausgegeben, so wird diese Ausgabe mit KDCLAST wiederholt.

Wird das Kommando KDCLAST nach Abschluss des Anmelde-Vorgangs eingegeben, dann zeigt openUTM nochmals den letzten Bildschirm des Anmelde-Vorgangs. Wird es nach der Rückkehr von einem eingeschobenen Vorgang eingegeben, wird der letzte Bildschirm des eingeschobenen Vorgangs ausgegeben.

KDCDISP, KDCLAST in FHS-DE-Formaten

Wenn die UTM-Kommandos KDCDISP und KDCLAST in FHS-DE-Formaten angegeben werden, dann werden sie von openUTM nur bearbeitet, wenn sie in das erste Eingabefeld oder in das UTM-Steuerfeld eines FHS-DE-Bildschirmformats angegeben werden. In dem Bildschirmformat darf kein Feld so belegt sein, dass als Folgeaktion ein FHS-DE-Zwischendialog geführt würde. Sind diese Bedingungen erfüllt, dann wird das Kommando von FHS-DE zur Bearbeitung an openUTM übergeben und als Folge die letzte Bildschirmausgabe wiederholt.

Wird eines der Kommandos dagegen in einer Dialog-Box (im FHS-DE-Zwischendialog) angegeben oder in einem Bildschirmformat, dem ein Zwischendialog folgt, dann führen KDCDISP und KDCLAST nicht zur Wiederholung der letzten Bildschirmausgabe durch openUTM. Zwischendialoge werden zuerst von FHS-DE bearbeitet. FHS-DE erkennt KDCLAST bzw. KDCDISP in diesem Fall nicht.

Im FHS-DE-Zwischendialog muss der Benutzer die Wiederholung der letzten Bildschirmausgabe mit der FHS-DE-Funktion RESHOW anfordern. Die Funktion RESHOW kann nur über eine K-Taste aufgerufen werden, die im zum Format gehörenden KEY-Format mit RESHOW belegt wird. Diese Taste darf in openUTM nicht mit SFUNC generiert werden. Siehe dazu auch Benutzerhandbuch „FHS Formatierungssystem für openUTM, TIAM, DCAM“.

Es wird empfohlen, die Wiederholung der letzten Bildschirmausgabe in FHS-DE-Formaten immer über die FHS-DE-Funktion RESHOW anzufordern.

8.6.5 KDCOFF - Abmelden von einer UTM-Anwendung

Der Benutzer kann sich durch Eingabe des UTM-Kommandos KDCOFF von der UTM-Anwendung abmelden. Dadurch wird die Verbindung zum Terminal abgebaut. Er kann danach von diesem Terminal erneut eine Verbindung zu einer UTM-Anwendung aufbauen. Meldet er sich während der Bearbeitung eines Vorgangs am Transaktionsende ab, wird die Bearbeitung unterbrochen. Sie kann nach dem nächsten Anmelden an die UTM-Anwendung wieder fortgesetzt werden, sofern der Benutzer mit RESTART=YES generiert ist.

KDCOFF BUT

Durch Eingabe von KDCOFF BUT kann der Benutzer sich so abmelden, dass die Verbindung zwischen Terminal und UTM-Anwendung bestehen bleibt. Er wird gleich zum erneuten Anmelden aufgefordert, bzw. der Anmelde-Vorgang wird gestartet.

Meldungen

Nach Eingabe von KDCOFF [BUT] reagiert openUTM mit einer der Meldungen:

`K019 KDCOFF von Anwendung <anwendungsname> akzeptiert`

Der Benutzer hat KDCOFF eingegeben, oder er hat in einer Anwendung ohne Benutzerkennungen KDCOFF BUT eingegeben. Das Terminal ist nicht mehr mit der UTM-Anwendung verbunden.

`K018 KDCOFF von Anwendung <anwendungsname> akzeptiert - Bitte KDCSIGN`

In einer Anwendung mit Benutzerkennungen und ohne Anmelde-Vorgang hat der Benutzer KDCOFF BUT eingegeben. openUTM fordert ihn zum erneuten Anmelden auf.

`K003 Das Kommando KDCOFF ist in dieser Situation nicht erlaubt`

Die Eingabe erfolgte nach einem PEND KP-Aufruf oder blockierenden Aufruf (z.B. PGWT) des Teilprogramms.

9 Programmaustausch im Betrieb

openUTM bietet Funktionen, mit denen Sie ein Anwendungsprogramm bzw. Teile eines Anwendungsprogramms im Betrieb austauschen können. Für den Austausch benutzt openUTM die Schnittstellen und Funktionen des BLS.

Voraussetzung für einen Programmaustausch ist, dass die Anwendung mit mindestens einer LOAD-MODULE-Anweisung generiert ist.

Im laufenden Betrieb können Sie mit Hilfe von UTM-Administrationsfunktionen Folgendes austauschen:

- Alle nicht-shareable Anwendungsteile, die nicht statisch gebunden sind
- Alle Anwendungsteile in einem Common Memory Pool, der für eine Benutzerkennung gültig ist (MPOOL..., SCOPE=GROUP). Dies setzt jedoch voraus, dass sich an den Common Memory Pool nicht mehrere UTM-Anwendungen angeschlossen haben.
- Die komplette Anwendung.
- Ein nicht-privilegiertes Subsystem im LLM-Format, bei dem der Private Slice als Lademodul mit LOAD-MODE=STARTUP | ONCALL generiert ist. Es ist zusätzlich zum KDCPROG-Kommando noch ein Systemadministrationskommando nötig. Es ist zu beachten, dass für nicht-privilegierte Subsysteme im OM-Format eine Konsistenzlücke existiert. Sehen Sie dazu die Hinweise im Abschnitt „Shared Code im Systemspeicher“.

Nicht im laufenden Betrieb austauschbar sind:

- Programmteile, die statisch zum Anwendungsprogramm gebunden wurden. Diese können nur mit dem gesamten Anwendungsprogramm ausgetauscht werden.
- Programmteile, die als shareable Programme in einen Common Memory Pool geladen wurden, der mit SCOPE=GLOBAL generiert ist.
- Programmteile in Common Memory Pools, die mit SCOPE=GROUP geladen wurden und an die sich mehrere UTM-Anwendungen unter derselben Benutzerkennung anschließen.
- Lademodule, in denen neben dem mit einer LOAD-MODULE-Anweisung generierten Lademodul zusätzliche benutzereigene Module per Autolink-Funktion geladen werden.
- Lademodule, die TCBs enthalten.
- Lademodule, die Laufzeitmodule enthalten. Diese können nur mit dem gesamten Anwendungsprogramm ausgetauscht werden.
- Das Lademodul, das das Administrationsteilprogramm KDCADM enthält.

Die UTM-Administrationsfunktionen zum Programmaustausch sind im openUTM-Handbuch „Anwendungen administrieren“ beschrieben.

9.1 Binden und Generieren

Damit Sie Anwendungsteile austauschen können, muss das Start-LLM als Bindelademodul (LLM) vorgebunden und in einer Programmbibliothek als Element vom Typ L bereitgestellt werden (siehe Abschnitt „Binden von LLMs“).

Die übrigen Programmteile müssen entweder ebenfalls als Bindelademodul (LLM) vorgebunden und in einer Programmbibliothek als Elemente vom Typ L oder Typ R bereitgestellt werden.

Austauschbare Lademodule müssen immer in einer LOAD-MODULE-Anweisung mit LOAD-MODE ungleich STATIC generiert sein. Falls sie in einen Common Memory Pool geladen werden, dann muss dieser mit MPOOL ,... SCOPE=GROUP generiert sein. Werden Teilprogramme oder Datenbereiche in ein Lademodul eingebracht, dann muss in den zugehörigen PROGRAM- oder AREA-Anweisungen der Operand LOAD-MODULE entsprechend versorgt werden.

9.2 Anwendungsteile austauschen

Einen Austausch von Anwendungsteilen müssen Sie explizit über das Administrationskommando KDCPROG anfordern. Dabei müssen Sie Angaben zur Version des neu zu ladenden Lademoduls machen. openUTM prüft die Zulässigkeit der Angaben und veranlasst den Programmaustausch. openUTM verifiziert beim Programmaustausch nicht, ob die mit KDCDEF definierte Zuordnung in der LOAD-MODULE-, AREA- und PROGRAM-Anweisung der tatsächlichen Aufteilung der Lademodule in den Bibliotheken entspricht.

Die Änderungen, die sich durch die Administrationsaktionen zum Programmaustausch in dem geladenen Anwendungsprogramm ergeben haben, werden von openUTM über das Ende des Anwendungslaufs hinaus gesichert; d.h. beim nächsten Start werden die Versionen der Lademodule geladen, die durch die Administrationsaktionen geändert wurden. Auch bei einer Änderungsgenerierung mittels KDCUPD können die Versionsnummern der ausgetauschten Lademodule in die neue KDCFILE übernommen werden, so dass bei dem nächsten Start der Anwendung wieder die zuletzt geladenen Module geladen werden.

Als Einzelmodul kann beim Programmaustausch nur ein Programmteil ausgetauscht werden, der mit **einem** Ladevorgang geladen wurde; d.h. als Teil kann jeweils **nur ein** LLM oder OM ausgetauscht werden. Es wird nur das bei openUTM generierte Modul ausgetauscht, nicht die gesamte Ladeeinheit, d.h. wenn die Ladeeinheit Teile des Laufzeitsystems enthält, die per Autolink dazugeladen wurden, so werden diese beim Austausch nicht entladen.

Beispiel

Das Lademodul A-LLM steht in der Bibliothek OWN-LIB und ist generiert mit

```
LOAD-MODULE A-LLM,LIB=OWN-LIB                                    -
                 ,VERSION=001                                    -
                 ,LOAD-MODE=STARTUP                              -
                 ,ALTERNATE-LIBRARIES=YES
```

A-LLM enthält beispielsweise ein Teilprogramm APU, das eine Funktion *bfunc* aufruft. Diese Funktion ist in B-LLM, das sich weder in A-LLM noch in der Bibliothek OWN-LIB befindet. In diesem Fall wird A-LLM beim Start des Anwendungsprogramms mit B-LLM per Autolink-Funktion geladen. Bei einem Austausch des A-LLM mit KDCPROG bleibt B-LLM mit *bfunc* im Speicher. Dies kann zu Inkonsistenzen führen, wenn B-LLM Unterprogramme von Anwendungsprogrammen und nicht ausschließlich Laufzeitmodule enthält. Enthält B-LLM auch Anwendungslogik, dann sollten A-LLM und B-LLM zu einem LLM gebunden werden.

Bei einem Austausch von einzelnen Lademodulen werden die Event-Exits SHUT bzw. START nicht durchlaufen. Diese werden nur dann aktiviert, wenn bedingt durch einen Programmaustausch in einer Task das gesamte Anwendungsprogramm beendet und neu geladen wird.

Der Austausch verläuft unterschiedlich, je nachdem, zu welchem Zeitpunkt (STARTUP oder ONCALL) und wohin (Common Memory Pool oder nicht) das Lademodul geladen wird.

9.2.1 Lademodul mit LOAD-MODE=STARTUP austauschen

Beim Austausch eines Lademoduls, das mit LOAD-MODE=STARTUP generiert wurde, wird der Programmlauf in den Tasks der Anwendung nicht beendet. Vielmehr wird das betroffene Lademodul entladen und anschließend eine neue Version dieses Lademoduls geladen. Diesen Programmaustausch können mehrere Tasks einer Anwendung gleichzeitig ausführen. Während des Austauschs sind in den Tasks der UTM-Anwendung unterschiedliche Stände des Anwendungsprogramms geladen. Jede Task der Anwendung führt den angeforderten Programmaustausch am Ende der Bearbeitung des aktuellen Auftrags durch. Der Programmaustausch ist abgeschlossen, wenn in jeder Task der Anwendung die neue Version des Lademoduls geladen ist.

Bis der Programmaustausch in allen Tasks abgeschlossen ist, darf kein weiterer Programmaustausch gestartet werden. Der Administrator kann sich mit dem Administrationskommando KDCINF SYSP darüber informieren, ob zur Zeit ein Programmaustausch läuft.

Die Versionsangaben zu altem und neuem Lademodul dürfen übereinstimmen.

Bei erfolgreichem Abschluss des Programmaustauschs erzeugt openUTM eine K074-Meldung, die nach SYSOUT ausgegeben wird. Die Meldung kann aber auch über ein MSGTAC-Programm auswertet werden, um dem Administrator diese Information zugänglich zu machen.

Muss der Programmaustausch durch openUTM abgebrochen werden, dann erzeugt openUTM eine K075-Meldung.

9.2.2 Lademodul mit LOAD-MODE=ONCALL austauschen

Soll ein Teil einer Anwendung ausgetauscht werden, der in einem Lademodul mit LOAD-MODE=ONCALL generiert wurde, dann wird bei der Bearbeitung des Administrationskommandos KDCPROG nur die neu zu ladende Versionsnummer des betroffenen Lademoduls in die UTM-Tabellen eingetragen.

Das Laden des Lademoduls der neuen Version wird von jeder Task der Anwendung erst dann veranlasst, wenn in der Task das nächste Mal ein Teilprogramm dieses Lademoduls aufgerufen wird. Diesen Programmaustausch können mehrere Tasks einer Anwendung gleichzeitig ausführen. Bis der angeforderte Programmaustausch von allen Tasks der UTM-Anwendung durchgeführt wurde, sind in den einzelnen Tasks unterschiedliche Stände des Anwendungsprogramms geladen. Es ist jedoch sichergestellt, dass jede Task den angeforderten Austausch durchführt, bevor erneut ein Teilprogramm aktiviert wird, das in dem auszutauschenden Lademodul enthalten ist.

Der Austausch eines ONCALL-Lademoduls wirkt nicht blockierend auf nachfolgende Kommandos zum Programmaustausch; d.h. der Administrator kann unmittelbar nach der Bearbeitung des Kommandos KDCPROG mit einem weiteren Kommando KDCPROG einen neuen Programmaustausch veranlassen. Sie dürfen jedoch den Inhalt der verwendeten Programmbibliotheken nach dem Administrationsaufruf nicht mehr verändern, da andernfalls der Programmaustausch zu Fehlern führen kann.

Stimmen die Versionsnummern von neuem und altem Lademodul überein, wird kein Programmaustausch durchgeführt.

9.2.3 Lademodul in einem Common Memory Pool austauschen

Lademodule, die ganz oder teilweise in Common Memory Pools liegen, können mit dem Kommando KDCPROG zunächst zum Austausch vorgemerkt werden.

Ausgetauscht werden diese Lademodule erst bei einem nachfolgenden Austausch der gesamten Anwendung, siehe unten.
Wird die Anwendung vor dem Anwendungstausch beendet, dann *bleiben* die Lademodule zum Austausch vorgemerkt. Beim nachfolgenden Neustart werden die neuen (vorgemerkten) Versionen geladen.

Bei Anforderung eines Programmaustauschs über KDCPROG müssen Angaben zur Version des neu zu ladenden Lademoduls gemacht werden. Die neue Version muss sich von der alten Version unterscheiden.

Teilprogramme, die sowohl shareable- als auch nicht-shareable-Teile enthalten, sollten als LLM gebunden werden, da sonst zwei Lademodule im OM-Format existieren, die nicht gleichzeitig ausgetauscht werden können (Konsistenzproblem!).

9.3 Gesamte Anwendung austauschen

Mit dem Kommando KDCAPPL PROG=NEW können Sie die gesamte Anwendung austauschen.

Der Austausch der gesamten Anwendung ist in den folgenden Fällen sinnvoll bzw. notwendig:

- Programmteile in Common Memory Pools sollen ausgetauscht werden.
- Programmteile, die mit LOAD-MODE=ONCALL generiert sind, sollen entladen werden.
- Programme in Common Memory Pools wurden dynamisch hinzugenommen, und diese Programme sollen für die Anwendung geladen werden.

Beim Austausch der gesamten Anwendung wird nacheinander jede Task der Anwendung entladen und anschließend neu geladen. Beim Neuladen werden die neuen Versionen der Lademodule geladen. Für Lademodule, die mit der Versionsangabe *HIGHEST-EXISTING generiert sind, wird die höchste verfügbare Version geladen. Um den Betrieb der Anwendung möglichst wenig zu stören, tauscht openUTM zu einer Zeit immer nur eine Task der Anwendung aus.

Der Administrator kann sich mit dem Kommando KDCINF SYSP darüber informieren, ob zur Zeit ein Programmaustausch läuft.

Bei erfolgreichem Abschluss des Programmaustauschs erzeugt openUTM eine K074-Meldung, die nach SYSOUT ausgegeben wird. Die Meldung kann auch über ein MSGTAC-Programm auswertet werden, um dem Administrator diese Information zugänglich zu machen.

Können einzelne Lademodule nicht geladen werden oder sind nicht alle Programme, entgegen den Angaben bei der UTM-Generierung, in den Lademodul gebunden, dann führt dies nicht zum Abbruch des Anwendungsaustausches. Beim späteren Aufruf eines nicht verfügbaren Programms erzeugt openUTM einen PEND ER.

Muss der Programmaustausch durch openUTM abgebrochen werden, dann erzeugt openUTM eine K075-Meldung. Die K075-Meldung enthält als Insert die TSN der Task, die den Anwendungsaustausch abgebrochen hat. In der SYSOUT-Datei dieser Task kann nach Meldungen des BLS und Meldungen von openUTM gesucht werden, die Hinweise auf den Grund des Abbruchs geben. Ein Grund für den Abbruch eines Anwendungsaustausches kann beispielsweise sein, dass eine AREA in einem neugeladenen Lademodul nicht verfügbar ist. Wird der Anwendungsaustausch abgebrochen, dann wird evtl. eine Task der UTM-Anwendung beendet. Die anderen Tasks der Anwendung laufen jedoch weiter. Die zum Austausch vorgemerkten Lademodule bleiben vorgemerkt, und sobald das Problem behoben ist, kann der Anwendungsaustausch erneut angestoßen werden.

9.4 Programme dynamisch hinzufügen

Per dynamischer Administration können u.a. Programme im Betrieb der Anwendung neu generiert werden. Näheres zur dynamischen Administration siehe openUTM-Handbuch „Anwendungen administrieren“.

Bevor diese Programme aufgerufen werden können, müssen sie zunächst geladen werden. Dazu muss das Programm zu dem zugeordneten Lademodul gebunden und mit einer neuen Version in der Programmbibliothek bereitgestellt werden, die in der LOAD-MODULE-Anweisung beim Generieren angegeben wurde.

Anschließend muss der Administrator dieses Lademodul durch das Kommando KDCPROG austauschen. Falls das Lademodul in einem Common Memory Pool liegt, muss die gesamte Anwendung ausgetauscht werden, d.h. zunächst wird das Modul zum Austausch vorgemerkt (z.B. mit KDCPROG) und anschließend wird die gesamte Anwendung ausgetauscht (z.B. mit KDCAPPL PROG=NEW).

10 Fehlertoleranz von openUTM

Fehlertoleranz heißt hier, dass eine UTM-Anwendung auch dann betriebsbereit bleibt, wenn in einzelnen Teilprogrammen Fehler auftreten, die openUTM zum Abbruch einer Transaktion zwingen. openUTM sorgt dann dafür, dass das Anwendungsprogramm beendet und neu geladen wird, so dass sich der Fehler nicht weiter ausbreitet und andere Benutzer der Anwendung und deren Daten dadurch nicht beeinträchtigt werden.

Beim Fehlerverhalten von openUTM unterscheidet man:

- Interne UTM-Fehler und Fehler in der Systemumgebung

  Sie führen zum abnormalen Beenden der Anwendung, ebenso wie das Administrationskommando KDCSHUT KILL oder das Absetzen eines KDCADMI-Aufrufs mit Operationscode KC_SHUTDOWN und Subcode KC_KILL. openUTM erzeugt für jeden Prozess der Anwendung einen UTM-Dump. Der UTM-Dump wird mit dem Tool KDCDUMP aufbereitet. Wie das geht, ist im openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“ beschrieben.
- Fehler im Anwendungsprogramm

  Dabei handelt es sich um Fehler in Teilprogrammen, die sich in zwei Gruppen einteilen lassen:
  - Fehler, die zum Neuladen des Anwendungsprogramms führen
  - Fehler, die eine Fortsetzung des Programms erlauben

10.1 Fehler, die openUTM erkennt

Ein Teilprogramm wird in folgenden Fällen durch openUTM abnormal beendet:

- Es wurde ein PEND ER oder FR programmiert.
- Ein UTM-Aufruf hat einen KDCS-Returncode >= 70Z geliefert. In diesem Fall setzt openUTM intern PEND ER.

In allen Fällen bricht openUTM also den Vorgang ab. Falls PEND FR programmiert war, veranlasst openUTM keine weitere Aktionen.

Falls der Vorgang durch PEND ER (per Programm oder intern) beendet wurde, erzeugt openUTM

- einen UTM-Dump mit REASON=PENDER, der nur die Daten des KDCROOT wiedergibt und
- einen Speicherauszug des Klasse 5- und Klasse 6-Speichers. Dieser kann mit dem BS2000-Dienstprogramm DAMP ausgewertet werden.

  Standardmäßig wird dieser Speicherauszug unterdrückt, da openUTM als ENTER-Prozess läuft. Möchten Sie Speicherauszüge des Klasse-5- und Klasse-6-Speichers erzeugt haben, geben Sie in der Startprozedur (siehe "Starten der Anwendung") vor dem Laden des Anwendungsprogramms das Kommando

  ```
  /MODIFY-TEST-OPTIONS DUMP=YES
  ```

Anschließend beendet openUTM das betroffene Anwendungsprogramm. Dadurch wird verhindert, dass Folgefehler wegen eines eventuell überschriebenen Anwendungsprogramms entstehen.

Mit einer Sprunganweisung in der Startprozedur lässt sich erreichen, dass das Anwendungsprogramm neu geladen wird und openUTM mit der gewünschten Anzahl Tasks weiterläuft.

> **ACHTUNG!**
>
> Fehlt die Sprunganweisung, wird die Task und im Fall der letzten Task die Anwendung beendet.

10.2 Fehler, die das BS2000-System erkennt und zu einem STXIT- Ereignis führen

In openUTM sind für die folgenden Ereignisklassen STXIT-Routinen definiert:

- PROCHK
- TIMER
- ERROR
- ABEND
- TERM

Bei Eintritt der STXIT-Ereignisse PROCHK, TIMER und ERROR ermöglicht openUTM dem Sprachanschluss-Modul des zuletzt aktiven Teilprogramms alternativ:

- Diagnoseinformationen auszugeben und das Teilprogramm abnormal zu beenden, oder
- das Teilprogramm definiert fortzusetzen (z.B. über ON-Condition in PL/I)

Tritt ein STXIT-Ereignis ABEND oder TERM ein, setzt openUTM PEND ER, beendet das Anwendungsprogramm mit TERM und schreibt Folgendes nach SYSOUT:

- die Registerinhalte zum Zeitpunkt des Ereignisses
- die Befehlszähler zum Unterbrechungszeitpunkt
- das Unterbrechungsgewicht.

Waren die zuletzt aktiven Teilprogramme COBOL-, Assembler- oder SPL-Teilprogramme, so werden alle STXIT-Ereignisse behandelt wie TERM und ABEND.

Zu STXIT-Ereignissen siehe BS2000-Handbuch „Makroaufrufe an den Ablaufteil“. Eine Tabelle zur Zuordnung von Ereignis zu Unterbrechungsgewicht finden Sie im openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“.

Benutzerdefinierte STXIT-Routinen

In den Teilprogrammen können den Ereignissen PROCHK und ERROR eigene STXIT-Ausgänge zugeordnet werden. Dabei müssen durch Aufrufe des STXIT-Makros mit dem Operanden STXDNEW eigene Verwaltungsblöcke für diese Ereignisse angelegt werden.

Diese benutzerdefinierten STXIT-Routinen **müssen** mit dem Makroaufruf EXIT CONTINU=YES beendet werden, damit openUTM seine eigene (und gleichzeitig letzte) STXIT-Routine derselben Ereignisklasse starten kann.

11 SAT-Protokollierung

Sicherheitsrelevante UTM-Ereignisse können mit der BS2000-Funktion SAT (**S**ecurity **A**udit **T**rail) protokolliert werden. SAT dient u.a. zur Beweissicherung bei unerlaubten Eindringversuchen und ermöglicht es sofort auf derartige Ereignisse zu reagieren (Alarmfunktion). Dadurch kann ein eventueller Schaden gering gehalten oder ganz vermieden werden.

Voraussetzung dafür sind die BS2000-Komponente SECOS und das Subsystem SATCP. openUTM bietet Ihnen die Möglichkeit, die SAT-Protokollierung von UTM-Ereignissen für Ihre Anwendung zu steuern: durch die KDCDEF-Generierung (MAX-Anweisung) und durch die UTM-SAT-Administration (siehe "UTM-SAT-Administrationskommandos").

Für eine UTM-Anwendung kann die SAT-Protokollierung durch die UTM-Generierung (MAX ...,SAT=ON) oder durch die UTM-SAT-Administration (KDCMSAT SAT= ON) eingeschaltet werden. Per UTM-SAT-Administration kann die Protokollierung im laufenden Betrieb jederzeit wieder ausgeschaltet werden (KDCMSAT SAT= OFF). Bei eingeschalteter SAT-Protokollierung wird eine Mindestprotokollierung durchgeführt.
Die Mindestprotokollierung umfasst die folgenden UTM-Ereignisse:

- An- und Abmelden einer Task bei der UTM-Anwendung
- Eingabe eines UTM-SAT-Administrationskommandos
- Austausch von Programmteilen mit Hilfe des BLS

Zusätzlich können weitere Ereignisse definiert werden. Die Protokollierung dieser Ereignisse kann Ereignis-spezifisch, Benutzer-spezifisch und Auftrags-spezifisch ein- und ausgeschaltet werden. Das Vordefinieren der zu protokollierenden Ereignisse nennt man Preselection (siehe "Preselection - zu protokollierende Ereignisse einstellen"). Die Preselection kann per UTM-Generierung und per UTM-SAT-Administration erfolgen.

Der Aufbau der SAT-Protokollsätze ist im Anhang "Aufbau der SAT-Protokollsätze" beschrieben.

> **ACHTUNG!**
>
> Der BS2000-Sicherheitsbeauftragte (BS2000-Benutzerkennung SYSPRIV bei Auslieferung) kann die SAT-Protokollierung unterdrücken. Siehe BS2000-Handbuch „SECOS“. Der SAT-Administrator der UTM-Anwendung muss sich daher mit dem BS2000-Sicherheitsbeauftragten absprechen. Bei nicht erfolgter Absprache wird die Meldung K126 ausgegeben, siehe openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen“.

11.1 Sicherheitsrelevante UTM-Ereignisse

Es sind elf sicherheitsrelevante UTM-Ereignisse definiert, die protokolliert werden können. Für jedes Ereignis wird ein Ergebnis (Result) protokolliert: Erfolg (Success) oder Misserfolg (Failure). Erfolgreich heißt z.B. beim Anmelden eines Benutzers, dass openUTM die angegebene Benutzerkennung und die Authentisierungsdaten akzeptiert hat.

Der folgenden Tabelle können Sie entnehmen, welche UTM-Ereignisse es gibt und bei welchem Ergebnis sie protokolliert werden können.

Name des Ereignisses	Bedeutung	Protokollierung möglich
TASK-ON	Anschließen einer Task an die UTM-Anwendung	Erfolg
TASK-OFF	Abmelden einer Task von der UTM-Anwendung	Erfolg
SIGN	Anmelden eines UTM-Benutzers	Erfolg / Misserfolg
CHANGE-PW	Ändern des Benutzerkennworts	Erfolg / Misserfolg
START-PU	Erzeugen eines Auftrags bzw. Start eines Teilprogrammlaufes	Erfolg / Misserfolg
END-PU	Beendigung eines Teilprogrammlaufes	Erfolg
DATA-ACCESS	Zugriff auf einen Vorgangs-übergreifenden UTM-Speicherbereich (ULS, GSSB, TLS)	Erfolg / Misserfolg
ADM-CMD	Ausführung eines Administrationsaufrufs	Erfolg / Misserfolg
SEL-CMD	Ausführung eines UTM-SAT-Administrationskommandos	Erfolg / Misserfolg
CHG-PROG	Austausch von Lademodulen durch BLS	Erfolg / Misserfolg

Hinweise

- Das Ereignis SEL-CMD wird als einziges auch dann protokolliert, wenn die SAT-Protokollierung für die UTM-Anwendung ausgeschaltet ist, sofern die Generierung des BS2000-Systems eine SAT-Protokollierung zulässt.
- Die Ereignisse TASK-ON, TASK-OFF und CHG-PROG werden immer protokolliert, wenn die Protokollierung für die UTM-Anwendung eingeschaltet ist (Mindestprotokollierung).
- Das Ereignis „Transaktionsende“ (END-PU mit TACIDEN=T oder C) wird immer protokolliert, wenn die Protokollierung für END-PU nicht explizit durch Ereignis-spezifische Preselection ausgeschaltet (OFF in der Anweisung SATSEL) und wenn für diese Transaktion mindestens ein anderes Ereignis protokolliert wurde.
- Die anderen Ereignisse aus der Tabelle werden erst protokolliert, wenn sie durch Preselection definiert wurden und die Protokollierung eingeschaltet ist.
- Alle UTM-Ereignisse können mit der ALARM-Funktion von SAT gekoppelt werden. Beim Auftreten des Ereignisses wird dann eine Meldung an der Konsole des BS2000-Rechners ausgegeben.

11.2 Preselection - zu protokollierende Ereignisse einstellen

Die Ereignisse SIGN, CHANGE-PW, START-PU, END-PU mit TACAID=P, DATA-ACCESS und ADM-CMD werden auch bei eingeschalteter Protokollierung nicht automatisch protokolliert. Die Protokollierung muss zusätzlich Ereignis-spezifisch eingeschaltet werden. Die Steuerung der Protokollierung dieser Ereignisse wird Preselection genannt. Die Protokollierung kann Ereignis-spezifisch, Benutzer-spezifisch und Auftrags-spezifisch gesteuert werden. Die Preselection kann per UTM-Generierung und per UTM-SAT-Administration erfolgen.

Wenn die Preselection-Werte per Administration eingestellt werden, dann gilt:

- Die Preselection-Werte für ereignisgesteuerte Protokollierung gelten nur für die Dauer des Anwendungslaufs. Bei jedem Anwendungsstart werden wieder die generierten Werte eingestellt.
- Die Preselection-Werte für Benutzer- und Auftrags-spezifische Protokollierung bleiben dauerhaft wirksam, auch bei UTM-F. Sie können mit dem inversen KDCDEF in eine neue UTM-Generierung übernommen werden.

11.2.1 Ereignisgesteuerte SAT-Protokollierung

Für die Ereignisse SIGN, CHANGE-PW, START-PU, END-PU, DATA-ACCESS und ADM-CMD kann die SAT-Protokollierung jeweils einzeln ein- und ausgeschaltet werden. Für das Ereignis DATA-ACCESS (Zugriff auf UTM-Speicherbereich) kann die Protokollierung für die Speichertypen GSSB, TLS und ULS jeweils einzeln gesteuert werden.

Sie können für jedes einzelne Ereignis angeben:

OFF	Das Ereignis wird nie protokolliert, auch wenn die Benutzer- bzw. Auftragsspezifische Protokollierung eingeschaltet ist.
SUCC	Das Ereignis wird im Erfolgsfall protokolliert.
FAIL	Das Ereignis wird protokolliert, falls es nicht erfolgreich war.
BOTH	Das Ereignis wird unabhängig vom Ergebnis protokolliert.
NONE	Keine ereignisgesteuerte Protokollierung

Die Preselection-Werte für die ereignisgesteuerte Protokollierung können Sie bei der UTM-Generierung einstellen durch die Anweisung:

```
USER username, ... ,SATSEL = ...
```

Per UTM-SAT-Administration können Sie die Preselection-Werte einstellen mit dem Kommando:

```
KDCMSAT SATSEL=...,EVENT=(...)
```

Diese Einstellung gilt dann nur für die Dauer des Anwendungslaufs.

Beispiel

Das Ereignis „Kennwort ändern“ (CHANGE-PW) soll protokolliert werden, im Erfolgsfall (Änderung wurde von openUTM akzeptiert) und bei Misserfolg. Das Absetzen eines Administrationskommandos soll bei einem Misserfolg protokolliert werden.

UTM-Generierung:

```
SATSEL BOTH,EVENT=CHANGE-PW

SATSEL FAIL,EVENT=ADM-CMD
```

Administration:

```
KDCMSAT SATSEL=BOTH,EVENT=CHANGE-PW

KDCMSAT SATSEL=FAIL,EVENT=ADM-CMD
```

11.2.2 Benutzergesteuerte SAT-Protokollierung

Für jeden einzelnen UTM-Benutzer kann definiert werden, ob die von ihm ausgelösten Ereignisse SIGN, CHANGE-PW, START-PU, END-PU, DATA-ACCESS, ADM-CMD und eventuell sicherheitsrelevante Ereignisse einer beteiligten Datenbank protokolliert werden sollen. Die Ereignisse werden für den Benutzer jedoch nicht protokolliert, wenn die SAT-Protokollierung für dieses Ereignis mit OFF ausgeschaltet ist.

UTM-Generierung: `SATSEL OFF,EVENT=...`
UTM-SAT-Administration: `KDCMSAT SATSEL=OFF,EVENT=...`

Sie können für jeden Benutzer angeben:

SUCC	Die Ereignisse, die der Benutzer auslöst, werden im Erfolgsfall protokolliert.
FAIL	Die Ereignisse, die der Benutzer auslöst, werden protokolliert, falls sie nicht erfolgreich waren.
BOTH	Die Ereignisse, die der Benutzer auslöst, werden unabhängig vom Ergebnis protokolliert.
NONE	Keine benutzergesteuerte Protokollierung

Die Preselection-Werte für die benutzergesteuerte Protokollierung können Sie bei der UTM-Generierung einstellen durch die Anweisung

`USER username,...,SATSEL=...`

Per UTM-SAT-Administration können Sie die Preselection-Werte einstellen mit dem Kommando:

`KDCMSAT SATSEL=...,USER=username`

Für SATSEL kann einer der Werte BOTH, SUCC, FAIL oder NONE angegeben werden. Diese Einstellung bleibt über das Anwendungsende hinaus erhalten, auch bei UTM-F.

11.2.3 Auftragsgesteuerte SAT-Protokollierung

Für jeden Transaktionscode kann definiert werden, ob die von dem zugehörigen Teilprogramm ausgelösten Ereignisse CHANGE-PW, START-PU, END-PU, DATA-ACCESS, ADM-CMD und evtl. sicherheitsrelevante Ereignisse einer beteiligten Datenbank protokolliert werden sollen. Zusätzlich wird auch das Erzeugen von Aufträgen dieses Transaktionscodes (START-PU) protokolliert. Ein Ereignis wird für den Transaktionscode jedoch nicht protokolliert, wenn die SAT-Protokollierung für dieses Ereignis mit OFF ausgeschaltet ist.

Sie können für jeden Transaktionscode angeben:

SUCC	Die Ereignisse des Teilprogrammlaufes werden im Erfolgsfall protokolliert.
FAIL	Die Ereignisse des Teilprogrammlaufes werden protokolliert, falls sie nicht erfolgreich waren.
BOTH	Die Ereignisse des Teilprogrammlaufes werden unabhängig vom Resultat protokolliert.
NONE	Keine auftragsgesteuerte Protokollierung

Die Preselection-Werte für die auftragsgesteuerte Protokollierung können Sie bei der UTM-Generierung einstellen durch die Anweisung:

```
TAC tacname,...,SATSEL=...
```

Per UTM-SAT-Administration können Sie die Preselection-Werte einstellen mit dem Kommando:

```
KDCMSAT SATSEL=...,TAC=tacname
```

Diese Einstellung bleibt über das Anwendungsende hinaus erhalten, auch bei UTM-F.

11.2.4 Preselection-Werte voreinstellen

Bei der UTM-Generierung können Sie Preselection-Werte angeben, ohne die SAT-Protokollierung einzuschalten. Die Angaben dienen in diesem Fall als Voreinstellung für die SAT-Protokollierung, die per UTM-SAT-Administration im laufenden Betrieb bei Bedarf eingeschaltet werden kann. Die definierten Werte können durch die UTM-SAT-Administration dann noch verändert werden. Änderungen für die ereignisgesteuerte Protokollierung gelten jedoch nur für die Dauer des aktuellen Anwendungslaufs.

Voreinstellung bei der UTM-Generierung:

```
MAX ...,SAT=OFF
SATSEL ...
TAC ...,SATSEL=...
USER ...,SATSEL=...
```

Einschalten per UTM-SAT-Administration:

```
KDCMSAT SAT=ON
```

11.2.5 Preselection-Werte verknüpfen

Werden für Ereignisse mehrere Preselection-Werte eingestellt (ereignis-, benutzer-, auftragsgesteuert), so ist das Ergebnis die Oder-Verknüpfung aus den einzelnen Preselection-Werten.

Die folgenden Tabellen zeigen die möglichen Kombinationen von SAT-Protokollierungsbedingungen.

Bedeutung der Spalten:

EVENT	Ereignis-spezifisch eingestellte Protokollierung (UTM-Generierung SATSEL ...; UTM-SAT-Administration KDCMSAT SATSEL=...,EVENT=...)
TAC	Auftrags-spezifisch eingestellte Protokollierung (UTM-Generierung TAC ...,SATSEL=...; UTM-SAT-Administration KDCMSAT SATSEL=...,TAC=...)
USER	Benutzer-spezifisch eingestellte Protokollierung (UTM-Generierung USER ...,SATSEL=...; UTM-SAT-Administration KDCMSAT SATSEL=...,USER=...)
Result	Ergebnis der Oder-Verknüpfung, d.h. der Kombination der Preselection-Werte.

Ist für ein Ereignis EVENT der Wert OFF (genereller Verzicht auf die Protokollierung des Ereignisses) generiert, so wird der Wert OFF als Ergebnis übergeben.

EVENT	SUCC															
USER	SUCC				FAIL				BOTH				NONE			
TAC	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE
Result	SUCC	BOTH	BOTH	SUCC	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	SUCC	BOTH	BOTH	SUCC

EVENT	FAIL															
USER	SUCC				FAIL				BOTH				NONE			
TAC	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE

Result	BOTH	BOTH	BOTH	BOTH	BOTH	FAIL	BOTH	FAIL	BOTH	BOTH	BOTH	BOTH	BOTH	FAIL	BOTH	FAIL

EVENT	BOTH															
USER	SUCC				FAIL				BOTH				NONE			
TAC	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE
Result	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH

EVENT	NONE															
USER	SUCC				FAIL				BOTH				NONE			
TAC	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE
Result	SUCC	BOTH	BOTH	SUCC	BOTH	FAIL	BOTH	FAIL	BOTH	BOTH	BOTH	BOTH	SUCC	FAIL	BOTH	NONE

Beispiel für die Verknüpfung der Preselection-Werte

Es werden folgende Preselection-Werte generiert:

Ereignis-spezifisch:

```
SATSEL FAIL,EVENT=(SIGN,TLS)
SATSEL SUCC,EVENT=(CHANGE-PW,GSSB,ADM-CMD)
SATSEL BOTH,EVENT=(START-PU,ULS)
SATSEL NONE,EVENT=END-PU
```

Benutzer-spezifisch:

```
USER BSPUSER,SATSEL=FAIL
```

Auftrags-spezifisch:

```
TAC BSPTAC,SATSEL=SUCC
```

Daraus ergeben sich als Ergebnis die folgenden Preselection-Werte. Die Preselection-Werte werden mit dem UTM-SAT-Administrationskommando KDCISAT abgefragt.

Ereignis-spezifisch mit dem Kommando KDCISAT:

SAT	USER	TAC	EVENT	RESULT
OFF			SIGN CHANGE-PW START-PU END-PU GSSB TLS ULS ADM-CMD	FAIL SUCC BOTH NONE SUCC FAIL BOTH SUCC

Benutzer-spezifisch mit dem Kommando KDCISAT USER=BSPUSER:

SAT	USER	TAC	EVENT	RESULT
OFF	BSPUSER		SIGN CHANGE-PW START-PU END-PU GSSB TLS ULS ADM-CMD	FAIL BOTH BOTH FAIL BOTH FAIL BOTH BOTH

Auftrags-spezifisch mit dem Kommando KDCISAT TAC=BSPTAC:

SAT	USER	TAC	EVENT	RESULT
OFF		BSPTAC	SIGN CHANGE-PW START-PU END-PU GSSB TLS ULS ADM-CMD	BOTH SUCC BOTH SUCC SUCC BOTH BOTH SUCC

11.3 Regeln der SAT-Protokollierung

- Die Ereignisse TASK-ON, TASK-OFF, CHG-PROG und mit Einschränkung END-PU (siehe Abschnitt „Sicherheitsrelevante UTM-Ereignisse“) werden genau dann protokolliert, wenn die SAT-Protokollierung für die UTM-Anwendung nicht ausgeschaltet ist.
- Alle anderen Ereignisse werden genau dann protokolliert, wenn die Protokollierung für das jeweilige Ereignis nicht ausgeschaltet ist und zusätzlich wenigstens eines der drei Kriterien Ereignis-spezifische, Benutzer-spezifische oder Auftrags-spezifische Preselection für dieses Ereignis erfüllt ist.
- Ein Ereignis wird von openUTM als erfolgreich protokolliert (SUCC), wenn es die UTM-Rechteprüfung erfolgreich durchlaufen hat und die Aktion von openUTM auch durchgeführt werden kann. Andernfalls wird das Ereignis als nicht erfolgreich (FAIL) protokolliert.
- UTM-Ereignisse, die durch KDCS-Aufrufe ausgelöst werden (z.B. DATA-ACCESS), werden nicht protokolliert, falls die Funktion wegen ungültigem Wert von KCOM, KCLA, KCLM usw. nicht ausgeführt werden kann (siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“). Die fehlerhafte Versorgung dieser Bereiche kann nur durch ein fehlerhaftes Teilprogramm erfolgen.
- UTM-Ereignisse, die durch KDCS-Aufrufe ausgelöst werden, werden als Misserfolg protokolliert, wenn die Versorgung von KCRN, KCUS oder KCLT fehlerhaft ist. Die fehlerhafte Versorgung von KCRN, KCUS und KCLT kann nicht nur durch ein fehlerhaftes UTM-Teilprogramm, sondern auch durch einen UTM-Benutzer verursacht werden (z.B. wenn dem Benutzer die Berechtigung für die aufgerufene Funktion fehlt).
- Anhand der SAT-Protokollsätze können Sie erkennen, ob ein Ereignis durch Rücksetzen der Transaktion nachträglich unwirksam wurde:

  Der Protokolldatensatz des betreffenden Ereignisses enthält die Identfikationsnummer der Transaktion, die das Ereignis ausgelöst hat (Protokollfeld UTMTAID).

  Über die Transaktions-Identifikation finden Sie den zugehörenden Protokollsatz „Transaktionsende“ (END-PU im Protokollfeld UTMSUBC). Dort ist im Feld UTMSTAT der Transaktionsstatus angegeben.

  Der Aufbau der Protokollsätze ist im Abschnitt „Aufbau der SAT- Protokollsätze“ im Anhang ("Aufbau der SAT-Protokollsätze") beschrieben.
- Der KDCS-Aufruf RSET (Rücksetzen der Transaktion bei Fortsetzung des Teilprogrammlaufes) löst implizit zuerst das Ereignis „Transaktionsende“ mit Transaktionsstatus „Rollback“ (OBJECT2= oder UTMSTAT= R) und dann das Ereignis „Beginn einer Transaktion“ (START-PU) aus.

11.4 Postselection - Protokollsätze auswerten

Die Protokollierung erfolgt durch SAT in eine besonders geschützte systemglobale Datei (SAT-Protokolldatei) auf der BS2000-Kennung SYSAUDIT.

Mit dem zu SECOS gehörende Dienstprogramm SATUT können nachträglich Datensätze aus der SAT-Protokolldatei ausgewählt und in eine Datei ausgegeben werden (Postselection). Die Auswahl der Protokollsätze kann über jedes protokollierte Feld erfolgen. Ein Datensatz wird nur übernommen, wenn die Datenfelder des zugehörigen Ereignisses bestimmte Regeln erfüllen. Für jeden Feldtyp kann eine der folgenden Bedingungen definiert werden:

- Der Wert des Feldes stimmt mit einem Element einer Liste überein.
- Der Wert des Feldes liegt innerhalb eines Intervalls.
- Das Datenfeld vom angegebenen Feldtyp ist im Datensatz vorhanden.

Weitere Informationen sind im BS2000-Handbuch „SECOS“ zu finden.

Der Aufbau der Protokollsätze ist im Anhang in "Aufbau der SAT-Protokollsätze" beschrieben.

11.5 Administration der SAT-Protokollierung

SAT protokolliert für UTM-Anwendungen ausgewählte sicherheitsrelevante UTM-Ereignisse. Bei der UTM-Generierung der Anwendung wird festgelegt, nach welchem Bearbeitungsresultat (Success, Failure) und nach welchen Kriterien (Ereignis-spezifisch, TAC-spezifisch oder Benutzer-spezifisch) die SAT-Protokollierung erfolgen soll. In diesem Kapitel sind die UTM-SAT-Administrationskommandos beschrieben, mit denen Sie die SAT-Protokollierung für Ihre UTM-Anwendung administrieren können.

Die UTM-SAT-Administrationskommandos sind jeweils eigene Transaktionscodes. Sie müssen daher bei der UTM-Generierung der Anwendung definiert werden. Die UTM-SAT-Administrationsfunktionen kann der UTM-SAT-Administrator nur über Dialog-TACs aufrufen. Die Namen der Transaktionscodes entnehmen Sie bitte der folgenden Tabelle:

Transaktionscode	Administrationsfunktionen
KDCMSAT	SAT-Protokollierung ein- und ausschalten SAT-Protokollierungswerte ändern Syntax des Kommandos abfragen
KDCISAT	Informationen über SAT-Protokollierungswerte anzeigen lassen Syntax des Kommandos abfragen

Sie können bei der UTM-Generierung festlegen, ob die SAT-Protokollierung automatisch bei jedem Start der Anwendung eingeschaltet werden soll oder nicht. Bei nicht eingeschalteter SAT-Protokollierung können trotzdem Kriterien für die SAT-Protokollierung generiert werden. Die Kriterien dienen dann als Vorbelegung für eine SAT-Protokollierung, die im laufenden Betrieb nach Bedarf ein- und ausgeschaltet werden kann.

Die generierten Protokollierungswerte können mit dem Administrationskommando KDCMSAT verändert werden. Die Änderung gilt für ereignisgesteuerte Protokollierung nur für die Dauer des aktuellen Anwendungslaufs. Für benutzer- und auftragsgesteuerte Protokollierung bleiben die Änderungen über das Anwendungsende hinaus erhalten.

Mit dem Administrationskommando KDCISAT können Sie sich die aktuell eingestellten Werte anzeigen lassen.

Die Eingaben der UTM-SAT-Administrationskommandos erfolgen im Line-Mode.

Die UTM-SAT-Administrationskommandos können nur von UTM-Benutzerkennungen aufgerufen werden, die UTM-SAT-Administrationsberechtigung haben. Die UTM-SAT-Administrationsberechtigung wird einer UTM-Benutzerkennung mit USER ...,PERMIT=SATADM oder PERMIT=(ADMIN,SATADM) bei der Generierung mit KDCDEF zugeordnet. Die UTM-Administrationsberechtigung (USER ..,PERMIT=ADMIN) allein berechtigt **nicht** zur UTM-SAT-Administration.

Damit Sie UTM-SAT-Administrationskommandos eingeben können, müssen bei der Generierung mit KDCDEF folgende Voraussetzungen erfüllt sein:

- Das Administrationsprogramm KDCSADM muss definiert sein (PROGRAM-Anweisung). Das Programm KDCSADM gehört zum Lieferumfang von openUTM.
- Die UTM-SAT-Administrationskommandos KDCMSAT und KDCISAT müssen als Transaktionscodes definiert sein (TAC-Anweisung mit PROGRAM=KDCSADM und SATADM=Y).

- Zumindest eine Benutzerkennung muss mit UTM-SAT-Administrationsberechtigung generiert werden (USER-Anweisung mit PERMIT=SATADM). Die Administrationsberechtigung kann gleichzeitig auch an mehrere Benutzerkennung vergeben werden. Die UTM-SAT-Administrationsberechtigung ist an den Benutzer gekoppelt und nicht an ein Terminal, d.h. man kann von jedem Terminal aus administrieren. Soll nur von bestimmten Terminals aus administriert werden, so wird dies mit den normalen Funktionen des Zugriffsschutzes realisiert (Operanden KSET= und LOCK=).

Beispiel

Die Transaktionscodes für die Administration werden mit der Steueranweisung TAC, das Administrationsprogramm wird mit der Steueranweisung PROGRAM des Generierungstools KDCDEF definiert:

```
PROGRAM KDCSADM,COMP=ILCS
:
TAC KDCMSAT ,PROGRAM=KDCSADM,SATADM=Y
TAC KDCISAT ,PROGRAM=KDCSADM,SATADM=Y
:
```

Die Operanden PROGRAM=KDCSADM und SATADM=Y der TAC-Anweisungen können bei Voreinstellung mit DEFAULT entfallen:

```
DEFAULT TAC PROGRAM=KDCSADM,SATADM=Y
```

Hinweise

- Jeder Zugriff auf den TAC KDCMSAT (außer KDCMSAT HELP) wird protokolliert, auch bei ausgeschalteter SAT-Protokollierung.
- Die UTM-SAT-Administration über Asynchron-Aufträge ist **nicht** möglich.

11.6 UTM-SAT-Administrationskommandos

Folgende Kommandos stehen zur Verfügung.

- KDCISAT - Informationen über SAT-Protokollierungswerte abfragen
- KDCMSAT - SAT-Protokollierung ändern

11.6.1 KDCISAT - Informationen über SAT-Protokollierungswerte abfragen

Mit dem Kommando KDCISAT kann sich der SAT-Administrator über die aktuell eingestellten Werte für die SAT-Protokollierung informieren oder die Syntax des Kommandos KDCISAT abfragen. KDCISAT liefert Informationen darüber, aus welchen UTM-Ereignisklassen die Ereignisse nach welchem Bearbeitungsergebnis (positiv oder negativ) protokolliert werden. Die Informationen können auch TAC- und/oder Benutzer-spezifisch abgefragt werden. Die UTM-Ereignisklassen sind bei der Beschreibung des Operanden EVENT bei der KDCMSAT-Anweisung aufgelistet ("KDCMSAT - SAT-Protokollierung ändern").

Bei jeder Abfrage mit KDCISAT wird ausgegeben, ob die SAT-Protokollierung ein- oder ausgeschaltet ist.

`KDCISAT`	`{ [ TAC=tacname ] [, USER=username] | HELP }`

KDCISAT ohne Operanden eingegeben:

> Es wird angezeigt, aus welchen Ereignisklassen Ereignisse protokolliert werden. Zu den einzelnen Ereignisklassen wird angegeben, ob erfolgreiche oder nicht erfolgreiche Ereignisse protokolliert werden (siehe Beispiel 1). Ausgegeben werden die Werte, die für alle Benutzer und für alle TACs gleich eingestellt sind. Das sind alle Werte der SAT-Protokollierung, die entweder in der SATSEL-Anweisung generiert oder mit dem Kommando KDCMSAT SATSEL=...,EVENT=(...) definiert wurden.

TAC=tac

> Es wird aus den einzelnen Ereignisklassen angezeigt, welche Ereignisse speziell für diesen TAC protokolliert werden (siehe Beispiel 2).

USER=user

> Es wird aus den einzelnen Ereignisklassen angezeigt, welche Ereignisse speziell für diesen Benutzer protokolliert werden.

TAC=tacname,USER=username

> Es wird für die einzelnen Ereignisklassen angezeigt, welche Ereignisse insgesamt für den angegebenen Benutzer und den angegebenen TAC protokolliert werden. Die Ausgabe setzt sich zusammen aus den eingestellten Ereignis-spezifischen Werten (KDCISAT ohne Operand), den TAC-spezifischen Werten (KDCISAT TAC=...) und den Benutzer-spezifischen Werten (KDCISAT USER=...) für die SAT-Protokollierung (siehe Beispiel 3).

HELP

> zeigt die Syntax dieses Kommandos an.

Ausgabe von KDCISAT

Alle Informationen werden auf das Terminal des SAT-Administrators ausgegeben. In der Spalte RESULT steht, ob Ereignisse protokolliert werden, die auf eine positive Ausführung oder auf eine negative Bearbeitung hinweisen etc.. Der in RESULT angegebene Wert ist das Ergebnis, das sich aus der Kombination der eingestellten SAT-Protokollierungsbedingungen ergibt. Das Ergebnis ist die Oder-Verknüpfung aus den für EVENT und gegebenenfalls USER und gegebenenfalls TAC eingestellten Protokollierungswerten (siehe Tabellen im Abschnitt „Preselection-Werte verknüpfen“).

Wird ein ungültiger USER oder TAC angegeben, so wird die Meldung `invalid TAC` oder `invalid USER` ausgegeben.

Beispiel 1

Bei der Eingabe von:

```
KDCISAT
```

erhalten Sie die folgende Ausgabe:

SAT	USER	TAC	EVENT [1]	RESULT
ON			SIGN CHANGE-PW START-PU END-PU GSSB TLS ULS ADM-CMD	BOTH BOTH NONE NONE BOTH BOTH BOTH BOTH

[1]Die Bedeutung der Ereignisklassen (EVENT) und der Werte von RESULT sind beim Kommando KDCMSAT (KDCMSAT - SAT-Protokollierung ändern) beschrieben.

Beispiel 2

Bei der Eingabe von:

```
KDCISAT TAC=tac1
```

erhalten Sie die folgende Ausgabe:

SAT	USER	TAC	EVENT	RESULT
ON		tac1	SIGN CHANGE-PW START-PU END-PU GSSB TLS ULS ADM-CMD	BOTH BOTH NONE NONE BOTH BOTH BOTH BOTH

Beispiel 3

Bei der Eingabe von:

```
KDCISAT TAC=tac2,USER=user2
```

erhalten Sie die folgende Ausgabe:

SAT	USER	TAC	EVENT	RESULT
ON	user2	tac2	SIGN CHANGE-PW START-PU END-PU GSSB TLS ULS ADM-CMD	BOTH BOTH NONE NONE BOTH BOTH BOTH BOTH

11.6.2 KDCMSAT - SAT-Protokollierung ändern

Bei der UTM-Generierung werden die Bedingungen für die SAT-Beweissicherung festgelegt. Es werden die Ereignisklassen angegeben, aus denen Ereignisse protokolliert werden sollen, und Benutzer-spezifisch, TAC-spezifisch oder Ereignis-spezifisch festgelegt, nach welchen Bearbeitungsergebnissen (erfolgreiche oder fehlerhafte Ausführung) protokolliert werden soll. Mit KDCMSAT kann der UTM-SAT-Administrator die generierten Werte modifizieren:

- Protokollierungswerte einstellen, jeweils mit einer eigenen Anweisung für EVENT, TAC oder USER.
- Protokollierung ein- oder ausschalten.

  Bei der UTM-Generierung können Sie SAT-Protokollierungswerte voreinstellen, ohne die SAT-Protokollierung selbst zu aktivieren. Mit KDCMSAT SAT=ON können Sie dann bei Bedarf die Protokollierung im laufenden Betrieb einschalten. Sie können die Protokollierung auch ohne Voreinstellungen bei der UTM-Generierung einschalten. Sie können dann mit KDCMSAT die Protokollierungswerte angeben.
- die Syntax dieses Kommandos abfragen.

Die Modifikationen für ereignisgesteuerte Protokollierung (SATSEL ...,EVENT=) und das Ein-/Ausschalten (SAT=ON /OFF) wirken nur bis zum Ende des Anwendungslaufs. Modifikationen für benutzer- und auftragsgesteuerte Protokollierung (SATSEL ...,TAC= und SATSEL...,USER=) bleiben über das Anwendungsende hinaus erhalten, auch bei UTM-F. Diese Werte können mit dem inversen KDCDEF in eine neue UTM-Generierung übernommen werden.

KDCMSAT	{ SATSEL={ BOTH | SUCC | FAIL | NONE | OFF }, { EVENT=(event1, ..., event*n*) | TAC=(tacname1, ..., tacname*n*) | USER=(username1, ..., username*n*) } *oder:* SAT={ OFF | ON } *oder:* HELP }

SATSEL =

steuert die Art der SAT-Protokollierung.

Der Wert von SATSEL ändert die generierte Protokollierungsart für das im folgenden Operanden angegebene Element der Anwendung.

Bei jedem Absetzen des Kommandos KDCMSAT können Sie jeweils die Einstellung für eines der Kriterien EVENT, TAC oder USER ändern.

BOTH — Es werden sowohl erfolgreiche als auch nicht erfolgreiche Ereignisse protokolliert.

SUCC — Erfolgreiche Ereignisse werden protokolliert.

FAIL — Nicht erfolgreiche Ereignisse werden protokolliert.

NONE Keine EVENT-, TAC- oder USER-spezifische Ereignisauswahl.

OFF Diese Einstellung ist nur für das Kriterium EVENT möglich. Für die in EVENT angegebenen Ereignisklassen werden dann keine Ereignisse protokolliert, auch wenn die SAT-Protokollierung bei der UTM-Generierung im USER- oder TAC-Kommando eingeschaltet wurde.

EVENT=(event1,...,event*n*)

Gibt die Liste der Ereignisklassen an, für die die Bedingungen der SAT-Protokollierung verändert werden sollen. Die folgenden Ereignisklassen können angegeben und beliebig kombiniert werden. Es dürfen maximal 8 Werte angegeben werden.

SIGN Anmelden eines Benutzers

CHANGE-**P**W Ändern des Passworts durch den Benutzer oder UTM-Administrator

START-PU Starten eines Teilprogrammlaufs oder Annahme eines Dialog- oder Asynchron-Auftrags

END-PU Ende eines Teilprogrammlaufs

GSSB Zugriff auf einen globalen sekundären Speicherbereich (GSSB)

TLS Zugriff auf einen Terminal-spezifischen Langzeitspeicher (TLS)

ULS Zugriff auf einen Benutzer-spezifischen Langzeitspeicher (ULS)

ADM-CMD Ausführung eines Administrationsaufrufs

TAC=(tacname1, ..., tacname*n*)

Gibt die Liste der Transaktionscodes an, für die die Bedingungen der SAT-Protokollierung verändert werden sollen. Sie können eine Liste von maximal 10 Transaktionscodes angeben.

USER=(username1, ..., username*n*)

Gibt die Liste der Benutzer (USER) an, für die die Bedingungen der SAT-Protokollierung verändert werden sollen. Sie können eine Liste von maximal 10 Benutzernamen angeben.

SAT=ON/OFF

SAT-Protokollierung ein-/ausschalten.

ON — Die SAT-Protokollierung wird eingeschaltet.

Mit SAT=ON können Sie die SAT-Protokollierung bei Bedarf während des Anwendungslaufs einschalten für Werte, die bei der UTM-Generierung voreingestellt oder mit einem vorhergehenden KDCMSAT-Kommando eingestellt wurden.

OFF — Die SAT-Protokollierung wird ausgeschaltet.

HELP

zeigt die Syntax dieses Kommandos an.

Ausgabe von KDCMSAT

Am SAT-Administrator-Terminal werden die alten und neuen Werte der SAT-Protokollierung ausgegeben, die mit diesem Kommando verändert bzw. ein- oder ausgeschaltet wurden.

Beispiel 1

Bei der Eingabe von:

```
KDCMSAT SATSEL=SUCC,EVENT=(START-PU,END-PU)
```

erhalten Sie die folgende Ausgabe:

SAT	EVENT	SATSEL	
		NEW	OLD
ON	START-PU END-PU	SUCC SUCC	FAIL FAIL

Für die Kriterien TAC und USER wird ein entsprechendes Bild ausgegeben.

Beispiel 2

Bei der Eingabe von:

```
KDCMSAT SAT=ON
```

erhalten Sie die folgende Ausgabe:

SAT	NEW	OLD
	ON	OFF

12 Accounting

openUTM stellt Accounting-Funktionen zur Verfügung, die es dem Betreiber einer UTM-Anwendung ermöglichen, den Benutzern der UTM-Anwendung die in Anspruch genommenen Betriebsmittel zu verrechnen. Das UTM-Accounting nutzt die Möglichkeiten des Betriebssystems, um die Abrechnungsdaten zu ermitteln und die Daten in der Abrechnungsdatei des Betriebssystems zu erfassen.

Mit dem Abrechnungssystem RAV des BS2000-Systems können diese Daten ausgewertet werden.

Detaillierte Informationen zum BS2000-Accounting finden Sie im BS2000-Handbuch „RAV - Rechenzentrums-Abrechnungsverfahren“.

Die Accounting-Funktionen, die das jeweilige Betriebssystem zur Verfügung stellt, können die Betriebsmittelauslastung und Leistung einer UTM-Anwendung nur als Ganzes erfassen. Wenn Sie die DV-Leistungen aber den einzelnen Benutzern der UTM-Anwendung zuordnen und in Rechnung stellen wollen, muss für das UTM-Accounting Folgendes berücksichtigt werden:

- Die Benutzer einer UTM-Anwendung werden durch Benutzerkennungen der UTM-Generierung repräsentiert und nicht durch Benutzerkennungen des Betriebssystems. Die von einem Benutzer in Anspruch genommenen Leistungen müssen also den einzelnen UTM-Benutzerkennungen zugeordnet werden können.
- In einer UTM-Anwendung ist eine Gruppe homogener Prozesse aktiv. Jeder Prozess bearbeitet nacheinander Aufträge für wechselnde Benutzer. Die innerhalb eines Prozesses in Anspruch genommenen Leistungen müssen deshalb pro aufgerufenen Service (d.h. für einzelne Teilprogrammläufe) ermittelt werden.
- Die Zeitbedingungen des OLTP-Betriebs erfordern eine Form der Leistungserfassung, die die Performance der Anwendung nicht beeinträchtigt.

Beim UTM-Accounting wird daher der Verbrauch an Betriebsmitteln erfasst, der von den einzelnen Teilprogrammen beansprucht wird. Damit kann der Betriebsmittelverbrauch dem Transaktionscode (TAC) des jeweiligen Teilprogramms und somit auch dem UTM-Benutzer, der den zugehörigen Service gestartet hat, zugeordnet werden.

Über die vom UTM-Accounting erfassten Leistungen hinaus gibt es einen Grundbedarf an Betriebsmitteln, der beim Ablauf einer UTM-Anwendung anfällt, aber nicht direkt einem Benutzer zugeordnet werden kann. Das sind:

- Plattenbelegung für KDCFILE-, SYSLOG- und USLOG-Dateien sowie
- CPU-Verbrauch und I/Os für
  - Starten und Beenden der UTM-Prozesse
  - Verbindungsbehandlung der zugeordneten Clients
  - LPUT-Behandlung (Übertragung auf USLOG-Datei)
  - Aufbereitung von Druckerausgaben

Sollen diese Leistungen bei der Abrechnung berücksichtigt werden, dann müssen Sie sie den Benutzern als Pauschale in Rechnung stellen.

12.1 Begriffsdefinitionen

In diesem Abschnitt werden einige Begriffe genauer erläutert, die für das UTM-Accounting relevant sind.

Benutzer im Sinne des UTM-Accounting

Der Benutzer einer UTM-Anwendung, für den eine Abrechnung erstellt werden soll, wird i.A. durch die UTM-Benutzerkennung repräsentiert.

Bei Anwendungen oder Clients, die sich nicht explizit mit einer echten Benutzerkennung angemeldet haben, wird der Name der Verbindungs-Benutzerkennung (TS-Anwendungen und UPIC-Clients), der LU6-Sessionname (LU6-Partner) bzw. der OSI-Association-Name (OSI TP-Partner) verwendet.

In UTM-Anwendungen ohne Benutzerkennungen ordnet openUTM die Leistungen, die von Terminals, UPIC-Clients oder TS-Anwendungen in Anspruch genommen wurden, ersatzweise den LTERM-Partnern zu.

Abrechnungsdatei

Alle Informationen, die das UTM-Accounting für die Benutzer-spezifische Abrechnung der Leistungen sammelt, schreibt openUTM in die Abrechnungsdatei des Betriebssystems.

Die Abrechnungsdatei wird vom BS2000-Systemadministrator verwaltet, der diese Abrechnungsdatei mit dem Tool RAV auswerten kann.

Betriebsmittel

Darunter werden folgende Leistungen zusammengefasst:

- Technische DV-Leistungen, insbesondere CPU-Verbrauch und I/Os
- Der Aufruf eines bestimmten Programms (Programmgebühr)

Kalkulationsphase

Die Kalkulationsphase dient als Orientierung für den Einsatz des Abrechnungsverfahrens.

In der Kalkulationsphase ermittelt openUTM für jedes aufgerufene Teilprogramm den Verbrauch der einzelnen Betriebsmittel und schreibt die Werte als Kalkulationssatz in die Abrechnungsdatei. Näheres ist in Abschnitt „Kalkulationsphase“ beschrieben.

Kalkulationssatz

Ein Kalkulationssatz ist ein Satz, den openUTM in der Kalkulationsphase pro Teilprogrammlauf in die Abrechnungsdatei schreibt. Er hat den Accounting-Satztyp UTMK. Die Datenfelder des Kalkulationssatzes UTMK sind im Anhang auf "Aufbau des Kalkulationssatzes" beschrieben.

Gewicht

Pro Betriebsmittel können Sie ein Gewicht (Faktor) festlegen. Dieses Gewicht gibt an, wie das Betriebsmittel im Vergleich mit anderen Betriebsmitteln zu werten ist. Der Verbrauch eines Betriebsmittels geht dann als Produkt „Gewicht * Betriebsmittelverbrauch“ in die Abrechnung ein. Die Gewichte für die einzelnen Betriebsmittel geben Sie bei der KDCDEF-Generierung in ACCOUNT an, siehe Abschnitt „Variante des Abrechnungsverfahrensfestlegen“.

Abrechnungsphase

In der Abrechnungsphase ermittelt openUTM für jedes Teilprogramm den Betriebsmittelverbrauch. Nach Beendigung eines Teilprogrammlaufs ermittelt openUTM anhand der generierten Gewichte und des generierten Festpreises die Summe der Verbrauchswerte.

Folgenden Leistungen werden berücksichtigt:

- CPU-Verbrauch
- Ein-/Ausgaben auf Platte
- Erzeugte Ausgabeaufträge für Drucker
- Festpreis für den Aufruf eines Teilprogramms

Das Ergebnis ist eine Anzahl von Verrechnungseinheiten, die auf den Benutzer-spezifischen Verrechnungseinheitenzähler aufaddiert wird.

openUTM schreibt erst dann einen Satz mit dem Inhalt dieses Zählers in die Abrechnungsdatei,

- wenn der Benutzer sich abmeldet und über keine andere Verbindung mehr bei der UTM-Anwendung angemeldet ist,
- oder wenn die Anwendung normal beendet wird
- oder wenn ein bestimmter Maximalwert überschritten wird. Diesen Maximalwert legen Sie bei der KDCDEF-Generierung mit ACCOUNT ...,MAXUNIT= fest.

Gewichte und Festpreise müssen Sie vor dem Start der Abrechnungsphase in die UTM-Generierung der Anwendung einbringen. Dabei können Sie wählen zwischen

- Festpreis-Abrechnung,
- verbrauchsorientierter Abrechnung,
- und einer Kombination der beiden Varianten.

Eine detaillierte Beschreibung der Abrechnungsphase finden Sie in Abschnitt „Abrechnungsphase“.

Die Abrechnungsphase des UTM-Accounting kann im laufenden Betrieb der UTM-Anwendung ein- und ausgeschaltet werden.

Abrechnungssatz

Ein Abrechnungssatz ist ein Satz, den openUTM in der Abrechnungsphase in die Abrechnungsdatei schreibt. Er hat den Accounting-Satztyp UTMA.
Die Datenfelder des Abrechnungssatz UTMA sind im Anhang auf "Aufbau des Abrechnungssatzes" beschrieben.

Verrechnungseinheiten

Verrechnungseinheiten sind das Produkt aus Verbrauch und Gewicht des jeweiligen Betriebsmittels. In der UTM-Abrechnung werden nur Verrechnungseinheiten gezählt. Mit Hilfe von RAV können sie in Kosten umgerechnet werden, die den Benutzern in Rechnung gestellt werden.

Verrechnungseinheitenzähler

openUTM führt in einer UTM-Anwendung pro Benutzer einen Verrechnungseinheitenzähler und akkumuliert dort den Verbrauch der Verrechnungseinheiten pro Benutzer.

Festpreis-Abrechnung

Mit dieser Variante des Abrechnungsverfahrens wird für einen Teilprogrammlauf eine konstante Anzahl von Verrechnungseinheiten in Rechnung gestellt. Diese Anzahl wird bei der Anwendungsgenerierung dem Transaktionscode zugeordnet. Die Gewichte der anderen Betriebsmittel sind dabei Null. Dabei können Sie einzelne Services auch kostenlos anbieten, z.B. Auskunftsfunktionen.

Verbrauchsorientierte Abrechnung

Mit dieser Variante des Abrechnungsverfahrens wird für einen Teilprogrammlauf der Verbrauch an Betriebsmitteln in Rechnung gestellt, der aktuell ermittelt wird. Die Verbrauchswerte für die Betriebsmittel werden entsprechend der generierten Gewichte gewichtet. Für den Aufruf eines Teilprogramms wird kein Festpreis berechnet.

Rechenzentrum-Abrechnungs-Verfahren (RAV)

Mit RAV können die Sätze weiterverarbeitet werden, die von UTM-Anwendungen in die BS2000-Abrechnungsdatei geschrieben werden. Dafür steht im RAV die Komponente RAV-UTM zur Verfügung.

12.2 Phasen des Accounting

Für die Durchführung des Accounting in UTM-Anwendungen sind folgende Schritte erforderlich:

- Kalkulationsphase
- Abrechnungsverfahren festlegen
- Abrechnungsphase
- Auswertung

12.2.1 Kalkulationsphase

Die Kalkulationsphase liefert Orientierungswerte, mit deren Hilfe Sie die einzelnen Betriebsmittel gewichten und Festpreise für die Inanspruchnahme eines Services festlegen können. Dabei ermittelt openUTM für jeden Teilprogrammlauf den Betriebsmittelverbrauch, stellt am Programmlaufende einen Kalkulationssatz vom Satztyp UTMK zusammen und schreibt ihn in die Abrechnungsdatei.

Die Kalkulationsphase kann im laufenden Betrieb jederzeit durch die UTM-Administration ein- und ausgeschaltet werden, z.B. um die generierten Gewichte zu überprüfen und gegebenenfalls bei der Neugenerierung zu aktualisieren.

Sie sollten jedoch beachten, dass openUTM bei eingeschalteter Kalkulationsphase nach jedem Teilprogrammlauf einen Satz in die Abrechnungsdatei schreibt. Dadurch wird die Performance der Anwendung belastet.

Kalkulationsphase einschalten

Die Kalkulationsphase können Sie bei der KDCDEF-Generierung oder per Administration einschalten, siehe auch openUTM-Handbuch „Anwendungen generieren“ und openUTM-Handbuch „Anwendungen administrieren“

- per KDCDEF-Anweisung ACCOUNT ACC=CALC
- oder per UTM-Administration
  - durch das Kommando KDCAPPL CALC=ON
  - oder über WinAdmin/WebAdmin
  - oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Im BS2000-Accounting muss der Systemverwalter den Satztyp UTMK einschalten.

Kalkulationsphase ausschalten

Die Kalkulationsphase können Sie nur per UTM-Administration ausschalten:

- durch das Kommando KDCAPPL CALC=OFF
- oder über WinAdmin/WebAdmin
- oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Daten eines Kalkulationssatzes

Ein Kalkulationssatz enthält folgende Daten:

- Zeitstempel des BS2000-Accounting
- Name der UTM-Anwendung
- Transaktionscode (TAC) des Teilprogramms
- CPU-Verbrauch im UTM-Prozess in msec
- CPU-Verbrauch im DB-System in msec, sofern das verwendete DB-System entsprechende Daten an openUTM liefert
- Anzahl I/Os im UTM-Prozess
- Anzahl I/Os im DB-System, sofern das DB-System entsprechende Daten liefert
- Länge der Eingabenachricht in Byte

- Länge der Ausgabenachricht in Byte
- Anzahl Ausgabeaufträge an Drucker
- Verrechnungseinheiten für LTAC-Aufrufe
- UTM-Benutzer, der den Service aufgerufen hat
- Name des LTERM-Partners, über den der Benutzer angemeldet ist
- Realzeit des Teilprogrammlaufs (msec)

Bei den Ausgabe-Nachrichten werden auch die mitgerechnet, die an ein Folgeteilprogramm gerichtet sind (z.B. nach PEND PR).

Aus den Kalkulationssätzen kann mit RAV eine Auswertung erzeugt werden, die den mittleren Betriebsmittelverbrauch pro TAC anzeigt. Sind mehrere UTM-Anwendungen im Einsatz, wird für jede UTM-Anwendung eine Auswertung erzeugt.

12.2.2 Variante des Abrechnungsverfahrens festlegen

Zunächst müssen Sie festlegen, ob Sie über Festpreis, über den Verbrauch oder über eine Kombination aus diesen beiden Varianten abrechnen. Ihre Entscheidung hängt davon ab, ob Sie bestimmte Leistungen der Anwendung mit festen Preisen anbieten oder den tatsächlichen Betriebsmittelverbrauch in Rechnung stellen wollen.

Festpreis-Abrechnung

Bei der Festpreis-Abrechnung kostet ein Teilprogrammlauf eine konstante Anzahl von Verrechnungseinheiten. Orientierungswerte dafür liefert die Kalkulationsphase. Daher ist eine Festpreis-Abrechnung die einfachste Lösung.

Die Anzahl der Verrechnungseinheiten geben Sie bei der KDCDEF-Generierung in der TAC-Anweisung beim Operanden TACUNIT an, siehe openUTM-Handbuch „Anwendungen generieren“.

```
TAC tacname,PROGRAM=progname,TACUNIT=anzahl_verrechnungseinheiten
```

Für jeden vom Benutzer aufgerufenen Transaktionscode wird der in TACUNIT angegebene Wert auf den Benutzer-spezifischen Verrechnungseinheitenzähler aufaddiert.

Bei der Festpreis-Abrechnung können Sie einige Services (z.B. Auskunftsfunktionen) auch kostenfrei zur Verfügung stellen. Dazu müssen Sie die zugehörigen Transaktionscodes wie folgt generieren:

```
TAC ... TACUNIT=0
```

Bei Verteilter Verarbeitung gilt Entsprechendes für die Anweisung LTAC und den Operanden LTACUNIT, siehe Abschnitt „Abrechnung bei verteilter Verarbeitung“.

Bei einer Festpreis-Abrechnung müssen Sie die Gewichte der Betriebsmittel in der KDCDEF-Anweisung ACCOUNT auf 0 setzen (=Standardwert).

Verbrauchsorientierte Abrechnung

Bei dieser Variante wird dem Benutzer der Verbrauch an Betriebsmitteln in Rechnung gestellt, der aktuell in der Abrechnungsphase ermittelt wird. Für die einzelnen Betriebsmittel müssen Sie Gewichte festlegen. Ein Gewicht ist ein Faktor, mit dem die verbrauchten Einheiten multipliziert werden. Bei der Wahl dieser Gewichte können Sie sich an den Verbrauchsdaten orientieren, die Sie in der Kalkulationsphase ermittelt haben.

Die Gewichte werden Anwendungs-spezifisch in der KDCDEF-Anweisung ACCOUNT festgelegt, d.h. sie gelten für alle Teilprogrammläufe.

Die Festlegung der Gewichte ist zwangsläufig subjektiv und hängt von der Installationsumgebung ab. Folgende Betriebsmittel können gewichtet werden:

- CPU-Verbrauch (ACCOUNT-Operand CPUUNIT)
- Ein-/Ausgaben auf Hintergrundspeicher (ACCOUNT-Operand IOUNIT)
- Druckerausgaben (ACCOUNT-Operand OUTUNIT)

Näheres siehe openUTM-Handbuch „Anwendungen generieren“.

Beispiel für die UTM-Generierung dieser Variante

```
ACCOUNT ACC=ON,CPUUNIT=15,IOUNIT=5,OUTUNIT=20
TAC tacname,PROGRAM=progname,TACUNIT=0
TAC ....
```

Pro Aufruf eines Transaktionscodes wird dann die folgende Summe auf den Verrechnungseinheitenzähler des Benutzers aufaddiert:

15 * CPU-Verbrauch + 5 * I/O-Verbrauch + 20 * Druckausgaben-Verbrauch

Kombination aus Festpreis- und verbrauchsorientierter Abrechnung

Sie können für Ihre Abrechnung die beiden obigen Varianten auch kombinieren, indem Sie für den Aufruf eines Transaktionscodes einen bestimmten Festpreis festlegen und zusätzlich den Verbrauch der Betriebsmittel (z.B. den CPU-Verbrauch) verrechnen.

In der Abrechnungsphase wird beim Aufruf eines Transaktionscodes die folgende Summe gebildet und auf den Verrechnungseinheitenzähler des Benutzers aufaddiert:

TACUNIT (Festpreis für den Aufruf des Teilprogramms)
+ CPUUNIT * CPU-Verbrauch + IOUNIT* I/O-Verbrauch
+ OUTUNIT * Druckausgaben-Verbrauch

Beispiel für die UTM-Generierung dieser Variante

```
ACCOUNT ACC=ON,CPUUNIT=15
TAC tacnam1,PROGRAM=progname1,TACUNIT=1
TAC tacnam2,PROGRAM=progname2,TACUNIT=2
```

12.2.3 Abrechnungsphase

In der Abrechnungsphase ermittelt openUTM die pro Teilprogrammlauf verbrauchten Betriebsmittel, berechnet daraus und aus den generierten Gewichten und Festpreisen eine gewichtete Summe. Diese Summe addiert openUTM auf den Verrechnungseinheitenzähler des UTM-Benutzers. Der Wert dieses Zählers ist im Abrechnungssatz enthalten, den openUTM in die Abrechnungsdatei schreibt.

openUTM schreibt immer dann einen Abrechnungssatz, wenn für den Benutzer eine bestimmte Anzahl von Verrechnungseinheiten aufaddiert sind oder wenn sich der Benutzer abmeldet und über keine andere Verbindung mehr bei der UTM-Anwendung angemeldet ist. Die Anzahl von Verrechnungseinheiten, bei der openUTM einen Abrechnungssatz schreibt, legen Sie bei der KDCDEF-Generierung in ACCOUNT MAXUNIT= fest. Dabei müssen Sie Folgendes beachten:

- Den Wert von MAXUNIT sollten Sie nicht zu klein wählen, da ein zu häufiges Schreiben von Abrechnungssätzen die Performance der Anwendung beeinträchtigen könnte.
- Den Wert von MAXUNIT sollten Sie nicht zu groß wählen, da bei einem Abbruch der Anwendung die Verrechnungseinheiten, die noch nicht in die Abrechnungsdatei geschrieben wurden, verloren gehen können (die Abrechnung unterliegt nicht der Transaktionssicherung).

Nachdem der Abrechnungssatz in die Abrechnungsdatei geschrieben wurde, werden der Verrechnungseinheitenzähler und der Zähler für die Anzahl der aufgerufenen TACs auf Null gesetzt.

Abrechnungsphase einschalten

Mit der KDCDEF-Steueranweisung ACCOUNT ACC=ON wird bei der UTM-Generierung die Abrechnung für die UTM-Anwendung eingeschaltet.

Zusätzlich kann die Abrechnungsphase auch im laufenden Betrieb durch die UTM-Administration ein- und ausgeschaltet werden

- durch das Kommando KDCAPPL ACCOUNT=ON
- oder durch WinAdmin/WebAdmin
- oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Im BS2000-Accounting muss der Systemverwalter den Satztyp UTMA einschalten.

Abrechnungsphase ausschalten

Die Abrechnungsphase können Sie nur per Administration ausschalten:

- durch das Kommando KDCAPPL ACCOUNT=OFF
- oder über WinAdmin/WebAdmin
- oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Daten des Abrechnungssatzes

Der Abrechnungssatz ist vom Satztyp UTMA. Er enthält folgende Daten:

- Zeitstempel des BS2000-Accounting
- Name der UTM-Anwendung
- UTM-Benutzerkennung
- Zeitpunkt der Anmeldung des Benutzers auf der aktuellen Verbindung
- Wert des Verrechnungseinheitenzählers
- Anzahl der aufgerufenen TACs mit TACUNIT > 0 seit der Anmeldung oder seitdem der letzte Satz geschrieben wurde

Sie können Kalkulationsdaten auch bei laufender Abrechnungsphase erfassen. Damit können Sie die Gewichte jederzeit überprüfen.

12.2.4 Auswertung

Das Ergebnis der Abrechnungsphase sind die Abrechnungssätze in der Abrechnungsdatei des BS2000-Systems. Sie können mit RAV ausgewertet werden. Die hierfür notwendigen Programme sind Bestandteil von RAV und werden zusammen mit diesem Produkt ausgeliefert. Sehen Sie hierzu auch Benutzerhandbuch „RAV (BS2000 /OSD) - Rechenzentrums-Abrechnungsverfahren“.

Der Aufbau der UTM-Abrechnungssätze ist im Anhang in Kapitel ''Aufbau des Abrechnungssatzes'' beschrieben.

12.2.5 Fehlersituationen

Kann das BS2000-Accounting auf Grund eines Fehlers einen Abrechnungssatz und/oder einen Kalkulationssatz nicht schreiben, beispielsweise weil nicht genügend Platz auf der Platte ist, erzeugt openUTM die Meldung K079 und beendet die Kalkulations- und/oder Abrechnungsphase. Ein Insert der Meldung K079 gibt die Ursache des Fehlers an. Die Anwendung läuft weiter.
Nach Behebung des Fehlers kann die Kalkulations- und/oder Abrechnungsphase durch die UTM-Administration (z. B. mit dem Administrationskommando KDCAPPL) wieder eingeschaltet werden.

12.3 Abrechnung bei verteilter Verarbeitung

Bei verteilter Verarbeitung kann im Prinzip jede der beteiligten Anwendungen Vorgänge in anderen Anwendungen starten. Die Abrechnung bei verteilter Verarbeitung ist vor allem dann sinnvoll, wenn die Rollen ungleich verteilt sind, d.h. eine Anwendung spielt ganz die Rolle des Auftraggebers und andere Anwendungen die Rollen der Auftragnehmer. Die Anwendungen werden in diesem Abschnitt daher als **Auftraggeber-Anwendung** und **Auftragnehmer-Anwendung** bezeichnet.

Die Auftraggeber-Anwendung nutzt Leistungen, die Teilprogramme in fernen Partner-Anwendungen (Auftragnehmer) für sie erbringen. In diesem Fall kann in der Auftraggeber-Anwendung der dabei anfallende Betriebsmittelverbrauch als Festpreis in Rechnung gestellt werden. Dazu ordnen Sie den LTACs in der Auftraggeber-Anwendung Verrechnungseinheiten als Festpreise zu. LTACs sind die Transaktionscodes, die in der Auftraggeber-Anwendung für einen Vorgang in einer Auftragnehmer-Anwendung definiert werden.

Näheres siehe openUTM-Handbuch „Anwendungen generieren“, Anweisung LTAC, Operand LTACUNIT.

Kalkulationsphase (Festlegen der Festpreise)

In der **Auftragnehmer-Anwendung** wird in der Kalkulationsphase der mittlere Betriebsmittelverbrauch der Teilprogramme ermittelt, die Services für die Auftraggeber-Anwendung erbringen. Anhand der hier ermittelten Verbrauchswerte können Sie die Festpreise festlegen, die den Benutzern der LTACs in der Auftraggeber-Anwendung berechnet werden sollen.

In der **Auftraggeber-Anwendung** zählt openUTM in einem Feld des Kalkulationssatzes die Verrechnungseinheiten, die bei den Aufrufen von LTACs anfallen.

Abrechnungsphase

In der **Auftragnehmer-Anwendung** werden alle Verbrauchswerte, die bei der Bearbeitung von Aufträgen für eine Auftraggeber-Anwendung anfallen, wie folgt zugeordnet:

- Bei LU6.1 den Sessions (LSES) zum Auftraggeber
- Bei OSI TP den Associations (OSI-LPAP ... ‚ASSOCIATION-NAME=), falls sich der OSI TP-Auftraggeber nicht unter einer echten Benutzerkennung angemeldet hat.

Die erbrachten Leistungen werden also der Auftraggeber-Anwendung insgesamt in Rechnung gestellt. Die Aufwände für die einzelnen Benutzer der Auftraggeber-Anwendung können nicht ermittelt werden.

In der **Auftraggeber-Anwendung** addiert openUTM beim Aufruf eines LTACs die Verrechnungseinheiten auf den Verrechungseinheitenzähler des Benutzers auf, die in der KDCDEF-Generierung in der LTAC-Anweisung angegeben wurden.

12.4 Einschränkungen

Bei der Nutzung des UTM-Accounting ist Folgendes zu beachten:

- Das Schreiben von Abrechnungsinformationen unterliegt nicht der Transaktionssicherung, deshalb können beim Abbruch einer Anwendung Verrechnungseinheiten verloren gehen. Der Maximalwert pro Benutzer kann per UTM-Generierung begrenzt werden.
- Für Anwendungen mit verteilter Verarbeitung wird in der Kalkulationsphase jeder LTAC-Aufruf mitgezählt. Es wird nicht berücksichtigt, ob nach der PEND-Verarbeitung eine Session belegt werden konnte oder nicht.
- Die Erfassung des Betriebsmittelverbrauchs beginnt vor dem Start eines Teilprogramms, sie endet in der Verarbeitung des PEND-Aufrufs. Die übrige Verarbeitungsleistung (Grundverbrauch) der UTM-Tasks wird den Benutzern nicht in Rechnung gestellt.
- Das Rücksetzen einer Transaktion hat folgende Auswirkungen: Alle Werte bis auf CPU und I/Os werden zurückgesetzt. Da openUTM die Verbrauchswerte in der PEND-Verarbeitung aufsummiert, kann eine Rücksetzaktion Verbrauchswerte nur dann zurücksetzen, wenn sie im aktuellen Teilprogrammlauf entstehen.
- Sind seit dem letzten Start der Anwendung für den Benutzer nur Asynchron-Aufträge verarbeitet worden, steht im Abrechnungssatz als Zeitpunkt der Anmeldung an die Anwendung der Wert Null.
- Für den Event-Exit VORGANG wird der Betriebsmittelverbrauch nur am Anfang des Vorgangs erfasst.
- Für den Event-Service BADTACS kann in der Abrechnungsphase kein Teilprogramm-Gewicht berücksichtigt werden.

13 Leistungskontrolle mit openSM2 und KDCMON

Die Performance einer UTM-Anwendung wird von verschiedenen Faktoren beeinflusst. Die Einflussfaktoren liegen zum einen im System-Umfeld einer UTM-Anwendung (Größe des Arbeitsspeichers, Leistungsfähigkeit der Peripherie) und zum anderen in der UTM-Anwendung selbst (Konfiguration der Anwendung und Aufbau der Teilprogramme). Im laufenden Betrieb einer Anwendung sollten Sie regelmäßig Leistungskontrollen durchführen, damit Sie rechtzeitig Hinweise auf Leistungsengpässe erhalten. Für UTM-Anwendungen stehen Ihnen die folgenden Instrumente zur Leistungskontrolle zur Verfügung:

- BS2000-Software Monitor openSM2
- UTM-Messmonitor KDCMON mit dem Auswertungstool KDCEVAL
- Informationsservices der UTM-Administration

Zusätzlich können Sie die Instrumente der Datenbanksysteme nutzen, z.B. SESCOS-Trace für SESAM/SQL.

Software Monitor openSM2

Der Software Monitor openSM2 des BS2000-Systems liefert statistische Daten über die Leistungen und die Auslastung der Betriebsmittel. Zusammen mit dem Subsystem UTM-SM2 ist es möglich, auch Anwendungs-spezifische Daten zu ermitteln und anzuzeigen. Diese Funktionen sollten Sie im laufenden Betrieb nutzen, um das Verhalten einer UTM-Anwendung zu überwachen und um Leistungsengpässe aufzudecken.

UTM-Messmonitor KDCMON

KDCMON zeichnet Informationen über den Ablauf von UTM-Anwendungen und Anwenderteilprogrammen auf. Zeichnen sich Leistungsengpässe ab, können Sie mit KDCMON Daten erfassen. Die gesammelten Daten müssen Sie mit dem Tool KDCEVAL auswerten. Anhand dieser Auswertung können Sie eine detaillierte Analyse durchführen. Siehe Kapitel "Auswertungslisten".

KDCMON ist damit ein wichtiges Werkzeug zur Beurteilung der Performance in einer UTM-Anwendung. KDCMON kann z.B. zur detaillierten Leistungsuntersuchung eingesetzt werden, wenn Messungen mit openSM2 bzw. Informationen der UTM-Administration auf Leistungsengpässe hinweisen.

Informationsservices der UTM-Administration

Einige Informationen zur Beurteilung der Anwendungsauslastung können Sie auch per UTM-Administration abfragen, z.B. über das Administrationskommando KDCINF oder über die grafischen Administrationstools WinAdmin/WebAdmin.

Das Kommando KDCINF STATISTICS liefert u.a. Daten über die Auslastung Ihrer UTM-Anwendung. Mit dem Kommando KDCINF STATISTICS können Sie allgemeine Statistikinformationen über die Anwendung abfragen und Kenngrößen für die Leistungskontrolle und die Beurteilung der Performance Ihrer UTM-Anwendungen im laufenden Betrieb ermitteln, wie beispielsweise die Auslastung der Anwendung, die Belegung des Pagepools, Anzahl der aktuell angemeldeten Benutzer, Anzahl der durchgeführten Dialog- oder Asynchron-Transaktionen pro Sekunde, offene Dialog- und Asynchron-Vorgänge usw..

Das Kommando KDCINF PAGEPOOL liefert weitere, detailliertere Daten über die aktuelle Belegung des Pagepools.

Ob die UTM-Anwendung Daten an openSM2 liefert, können Sie mit dem Administrationskommando KDCINF SYSPARM abfragen. Sehen Sie dazu im openUTM-Handbuch „Anwendungen administrieren“.

Wenn Sie die UTM-Anwendung mit dem grafischen Administrationsarbeitsplatz WinAdmin oder WebAdmin administrieren, können die Statistikdaten auch grafisch dargestellt werden.

13.1 Messdatenerfassung mit openSM2

Der Software-Monitor openSM2 in BS2000-Systemen erfasst statistische Daten über Leistung und Auslastung der Betriebsmittel.

openUTM kann Daten über UTM-Anwendungen an openSM2 liefern, die für eine erste Beurteilung der Performance wichtig sind. Die Menge der Daten ist unabhängig von der Größe der Konfiguration. Die Daten zeigen das Verhalten der gesamten Anwendung auf.

Zur Auswertung der von openSM2 erfassten Messdaten dient die openSM2-Komponente SM2R1, sowie die openSM2-Komponente SM2-PA zur Auswertung Benutzer-spezifischer Messwertdateien. openSM2 erfasst Messwerte und gibt sie auf Anforderung direkt am Terminal für eine Echtzeitüberwachung (online) aus. Zusätzlich sammelt openSM2 die Messdaten in einer Datei, die gespeicherten Daten können auf Anforderung zu einem späteren Zeitpunkt ausgewertet werden (offline).

Für die Leistungskontrolle einer UTM-Anwendung stehen Ihnen auch andere Funktionen von openSM2 zur Verfügung, z.B. globale Prozess-Auswertung, oder Vermessung der Prozesse einer UTM-Anwendung und Programmanalyse durch SM2-PA.

Voraussetzungen für die UTM-Messdatenerfassung durch openSM2

Damit openUTM Daten an openSM2 liefern kann, und openSM2 UTM-Daten sammeln, speichern und aufbereiten kann, müssen die im Folgenden beschriebenen Voraussetzungen erfüllt sein.

- Das Subsystem UTM-SM2 muss installiert und geladen sein.

  Für die Datenlieferung von openUTM an openSM2 ist das Subsystem UTM-SM2 notwendig. UTM-SM2 dient als Kommunikationsbaustein zwischen den Prozessen der UTM-Anwendung und openSM2. Es ist als eigenes Subsystem realisiert und ist in BS2000-GA (Grundausbau) enthalten. Wie das Subsystem UTM-SM2 vom BS2000-Systemverwalter zu installieren ist, ist im Anhang auf "Subsystem UTM-SM2" beschrieben.

  UTM-SM2 kann wie folgt geladen werden:

  1. Durch den BS2000-Systemverwalter mit dem Kommando:

     ```
     /START-SUBSYSTEM SUBSYSTEM-NAME=UTM-SM2
     ```

  2. Automatisch beim Start der Anwendung, wenn diese mit MAX SM2=ON generiert ist.
  3. Beim Einschalten der Datenlieferung durch die Administration, siehe unten. Dazu muss die Anwendung mit MAX SM2=OFF oder MAX SM2=ON generiert sein.

  Bei Bedarf kann das Subsystem UTM-SM2 mit dem Kommando STOP-SUBSYSTEM auch im laufenden Betrieb entladen werden. Das kann z.B. zum Austausch des Subsystems erforderlich werden. openSM2 und die UTM-Anwendungen beenden dann ihre Zusammenarbeit mit UTM-SM2.

  Nach dem Neuladen muss die Datenlieferung wieder explizit für jede UTM-Anwendung eingeschaltet werden.

- Die Datenlieferung von openUTM an openSM2 muss in der UTM-Anwendung generiert sein.

  Damit openUTM Daten über eine UTM-Anwendung an openSM2 liefern kann, muss dies bei der UTM-Generierung der Anwendung angegeben werden. Dazu geben Sie in der MAX-Anweisung im Operanden SM2 den Wert ON oder OFF an.

  Wird MAX...,SM2=ON angegeben, so wird die Datenlieferung an openSM2 beim Start der Anwendung eingeschaltet. Sie kann dann im laufenden Betrieb bei Bedarf per UTM-Administration aus- und wieder eingeschaltet werden.

  Wird MAX...,SM2=OFF angegeben, so ist die Datenlieferung an openSM2 für diese Anwendung erlaubt. Sie muss aber im laufenden Betrieb durch die UTM-Administration explizit eingeschaltet werden.

  Wenn MAX ...,SM2=NO generiert wird, so liefert openUTM für diese Anwendung keine Daten an openSM2. Die Datenlieferung kann dann auch nicht von der UTM-Administration veranlasst werden.

- Einschalten der Datenlieferung an openSM2 durch die UTM-Administration.

  Mit dem Kommando KDCAPPL SM2=ON kann der UTM-Administrator die Datenlieferung an openSM2 einschalten, wenn dies in der UTM-Generierung vorgesehen wurde. Mit KDCAPPL SM2=OFF wird die Datenlieferung ausgeschaltet.

  Mit dem Kommando KDCINF SYSPARM kann der UTM-Administrator feststellen, ob die Anwendung Daten an openSM2 liefern darf und ob sie aktuell Daten liefert.

  Das Ein- und Ausschalten der Datenlieferung an openSM2 ist auch über die „Programmschnittstelle Administration“ oder über die grafischen Administrationstools WinAdmin/WebAdmin möglich.

- Der openSM2-Verwalter (privilegierter openSM2-Benutzer) muss das Sammeln von UTM-Daten durch openSM2 starten.

  Der openSM2-Verwalter muss mit der Anweisung

  ```
  START-MEASUREMENT-PROGRAM TYPE=*UTM
  ```

  veranlassen, dass openSM2 Daten zu UTM-Anwendungen sammelt und auswertet. Zum Ausschalten gibt es analog dazu die STOP-MEASUREMENT-PROGRAM-Anweisung. Ist das Subsystem UTM-SM2 geladen, so kann von openSM2-Seite das Sammeln von UTM-Daten jederzeit eingeschaltet werden, unabhängig davon, ob von UTM-Seite die Datenlieferung eingeschaltet ist oder nicht. openSM2 kann mit der Verarbeitung der Daten jedoch erst beginnen, wenn in der jeweiligen UTM-Anwendung die Datenlieferung eingeschaltet wird.

Ausgabe und Auswertung der Messdaten

Der openSM2-Benutzer kann sich die von openUTM gelieferten Daten auf einem openSM2-Bildschirm (UTM-Report bzw. UTM-Application-Report) online anzeigen lassen. openSM2 speichert die Daten auch in einer systemglobalen Messwertdatei. Sie können sich die Daten somit auch später von SM2R1 auswerten lassen.

Der openSM2-Bildschirm „UTM-REPORT“ enthält eine tabellarische Übersicht von Daten zu allen UTM-Anwendungen, die aktuell Daten an openSM2 liefern. Es wird eine Zeile pro UTM-Anwendung ausgegeben.

Eine periodische Ausgabe des openSM2-Bildschirms „UTM-REPORT“ erhalten Sie mit der Anweisung

```
REPORT UTM
```

Aktuelle Messwerte zu einer oder mehreren ausgewählten UTM-Anwendungen liefert der openSM2-Bildschirm „UTM-APPLICATION-REPORT“.

Der openSM2-Bildschirms „UTM-APPLICATION-REPORT“ wird zusammen mit dem openSM2-Bildschirm "UTM-REPORT" periodisch für alle Anwendungen ausgegeben, die Sie zuvor ausgewählt haben. Die Anweisung zum Auswählen der Anwendungen lautet:

SELECT-UTM-APPLICATION (application1,application2,...)

Für *application1*,*application2*,... geben Sie die Namen der UTM-Anwendungen an, deren Verhalten Sie online verfolgen wollen. Sie müssen für *application1,..* jeweils den in der MAX-Anweisung generierten Namen der Anwendung angeben.

Die Bedeutung der ausgegebenen Daten ist im Benutzerhandbuch „openSM2“ erklärt, die Begriffe halten sich an den Sprachgebrauch von openUTM.

Neben der Möglichkeit, die Messdaten online zu verfolgen, gibt es folgende Auswertungen mit SM2R1 von Daten aus UTM-Anwendungen:

- die Reports 128 bis 133
- SUMMARY UTM

Sehen Sie dazu das Benutzerhandbuch „openSM2“.

13.2 UTM-Messmonitor KDCMON

KDCMON ist eine funktional eingeschränkte Variante von COSMOS. COSMOS ist ein Werkzeug, das auf BS2000-Systemen zur Leistungskontrolle eingesetzt wird. Mit KDCMON werden nur UTM- und bestimmte DB-Ereignisse aufgezeichnet. KDCMON und COSMOS können zu gleicher Zeit an einer Anlage ablaufen, ebenso können openSM2 und KDCMON zusammen eingesetzt werden.

KDCMON kann im laufenden Betrieb eingeschaltet und nach der gewünschten Messdauer wieder abgeschaltet werden. Die Daten können auf Band oder Platte geschrieben werden. Bei größeren Datenmengen sollten die Daten auf Bändern erfasst werden; dadurch können Stausituationen bei der Datenerfassung vermieden werden.

Zur Auswertung der von KDCMON erfassten Daten stehen die Tools KDCPMSM und KDCEVAL zur Verfügung:

- KDCPMSM setzt die von KDCMON erfassten Daten um und sortiert sie
- KDCEVAL erzeugt aus den umgesetzten Daten Auswertungslisten

KDCMON ist auch im Parallelbetrieb von openUTM-Versionen einsetzbar, d.h. KDCMON kann Daten von UTM-Anwendungen aufzeichnen, die unter verschiedenen openUTM-Versionen im gleichen BS2000-System ablaufen.

13.2.1 Erfassung starten und beenden

Die Erfassung der Daten wird in zwei Schritten gestartet:

- zuerst müssen Sie KDCMON starten
- danach schalten Sie die Datenaufzeichnung für die UTM-Anwendungen ein, die überprüft werden sollen

KDCMON ist ein eigenständiges Subsystem im BS2000-System und ist im BS2000-Grundausbau enthalten. KDCMON muss vom Systemverwalter installiert und geladen werden. Sehen Sie dazu auch Abschnitt „Subsystem KDCMON“.

KDCMON starten

Vor dem Start von KDCMON müssen Sie zunächst eine Datei einrichten, in die KDCMON die erfassten Daten schreiben soll. Dazu müssen Sie folgende Kommandos absetzen:

- für die Erfassung auf Platten-Dateien

```
/SET-FILE-LINK LINK-NAME=KDCMON,FILE-NAME=kdcmonfile -
/     ,ACCESS-METHOD=UPAM
/MODIFY-FILE-ATTRIBUTES FILE-NAME=kdcmonfile         -
/     ,SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE          -
/      (PRIMARY-ALLOCATION=xxx,SECONDARY-ALLOCATION=yyy))
```

  xxx und *yyy* müssen Vielfache von 12 sein (sonst DVS-Fehler).

- für die Erfassung auf Band-Dateien

```
/SET-FILE-LINK LINK-NAME=KDCMON,FILE-NAME=kdcmonfile  -
/     ,ACCESS-METHOD=BTAM
/MODIFY-FILE-ATTRIBUTES FILE-NAME=kdcmonfile          -
/     ,SUPPORT=*TAPE(VOLUME=xxxxxxx,DEVICE-TYPE=type)
```

> ¡ Blockgröße und die Satzlänge dürfen für die Datei nicht angegeben werden.

Danach können Sie KDCMON unter der Kennung TSOS starten. Das Programm zum Starten von KDCMON wird in der Datei SYSPRG.KDCMON.*nnn* bereitgestellt:

```
/START-EXECUTABLE-PROGRAM FROM-FILE=$userid.SYSPRG.KDCMON.nnn
```

Das Versionskennzeichen *nnn* steht für die BS2000-Version, in der KDCMON abläuft. Sehen Sie dazu im Anhang den Abschnitt „Subsystem KDCMON“.

> ¡ Sie können KDCMON auch über das SDF-Kommando START-KDCMON starten, siehe Abschnitt „UTM-Tools über eigene SDF-Kommandos starten“.

Das Programm KDCMON erwartet folgende Steuerparameter:

BUFPAG=size

gibt die Größe des Puffers in KDCMON in Einheiten von 4 KB an.

erlaubte Werte: 1 bis 7

Standardwert: 2 (= empfohlener Wert)

BUFFER=number

gibt die Anzahl der Puffer in KDCMON an.

erlaubte Werte: 2 bis 128

Standardwert: 4 (= empfohlener Wert)

{TIME= mmm | START | BREAK}

bestimmt die Laufzeit von KDCMON.

Die Angabe einer der Alternativen TIME=*mmm*, START oder BREAK beendet die Eingabe der Steuerparameter.

TIME=mmm	Laufzeit von KDCMON in Minuten. Nach der angegebenen Zeit beendet sich KDCMON. Minimalwert: 1 Maximalwert: 150
START	KDCMON läuft 30 Minuten. Entspricht der Angabe TIME=30 Standardwert: START
BREAK	KDCMON läuft so lange, bis RESUME eingegeben wird.

> **ACHTUNG!**
>
> KDCMON darf maximal 150 Minuten laufen, sonst kann das Auswertungstool KDCEVAL die Daten nicht auswerten.

Das Programm SYSPRG.KDCMON.*nnn* darf während einer Datenerfassung nicht beendet werden, anderenfalls ist das Subsystem KDCMON nicht mehr verfügbar und muss entladen werden.

KDCMON beenden

KDCMON beendet sich nach Ablauf der vorgegebenen Zeit für Parameter TIME=*mmm* und START, oder nach der Eingabe von RESUME bei BREAK. Das Subsystem KDCMON bleibt aber weiterhin geladen und kann mit dem DSSM-Kommando

```
/STOP-SUBSYSTEM KDCMON
```

entladen werden, wenn zu diesem Zeitpunkt keine Datenerfassung läuft. Soll KDCMON entladen werden, wenn die Datenaufzeichnung in einer der UTM-Anwendungen noch nicht abgeschlossen ist, dann muss

```
/STOP-SUBSYSTEM KDCMON,FORCED=YES
```

angegeben werden.

Datenerfassung einschalten und ausschalten

Nach dem erfolgreichen Start von KDCMON können Daten von allen UTM-Anwendungen aufgezeichnet werden, die auf dem jeweiligen Rechner laufen.

KDCMON erfasst auch BCAM-Wartezeiten: BCAM erfasst bei eingeschaltetem KDCMON für Eingabe-Nachrichten die Wartezeit, die diese beim Transportsystem BCAM verbringen, bis openUTM sie bei BCAM abholt. Bei Verbindungen mit strengem Dialog werden für alle Eingabe-Nachrichten die Wartezeiten erfasst. Bei anderen Verbindungen wird in Hochlastsituationen eine statistische Auswahl getroffen. openUTM übernimmt diese Wartezeit pro Nachricht von BCAM und schreibt sie in die KDCMON-Sätze. KDCEVAL ermittelt daraus den maximalen, minimalen und den mittleren Wert (Angabe in Sekunden mit einer Genauigkeit in Millisekunden). Diese Werte werden in der Liste WAIT (siehe "WAIT: WAITING TIMES") ausgegeben.

Mit Hilfe der UTM-Administration wird gesteuert, von welchen UTM-Anwendungen KDCMON Daten aufgezeichnet. Die Datenerfassung inklusive Erfassung der BCAM-Wartezeiten können Sie mit dem folgenden Administrations-Kommando ein- und ausschalten:

```
KDCDIAG KDCMON={ ON | OFF }
```

Diese Administrationsfunktion steht auch an der Programmschnittstelle KDCADMI und über die grafischen Administrationstools WinAdmin/WebAdmin zur Verfügung.

Man kann während eines KDCMON-Laufes für eine Anwendung die Erfassung mehrfach ein- und wieder ausschalten. Dabei sind bis zu 10 Erfassungs-Zeiträume möglich.

Der UTM-Administrator kann jederzeit mit Hilfe des Kommandos

```
KDCINF SYSPARM
```

feststellen, ob Daten erfasst werden oder nicht.

Wenn openUTM beim Einschalten feststellt, dass die KDCMON-Funktion nicht verfügbar ist, dann wird folgende Meldung mit dem Standard-Ziel SYSLOG ausgeben:

```
K080 KDCMON-Funktion nicht verfuegbar
```

Mögliche Ursache:
Das Subsystem KDCMON wurde nicht gestartet oder ist nicht unter $TSOS installiert worden.

Wenn openUTM bei laufender Erfassung feststellt, dass die KDCMON-Funktion nicht mehr verfügbar ist, dann schaltet openUTM die Datenerfassung aus und protokolliert dies ebenfalls mit der Meldung K080.

openUTM protokolliert das Ein- und Ausschalten der Erfassung der BCAM-Wartezeit mit der Meldung K146; Standardziel der Meldung ist SYSLOG. Die Meldung K146 gibt openUTM auch aus, wenn beim Lesen der BCAM-Wartezeiten ein Fehler auftritt. Als Diagnoseunterlage erzeugt openUTM dann einen UTM-Dump mit Dump-Fehlercode ASIS70, UMES02 oder WAIT61. Die Anwendung läuft dann ohne Erfassung der BCAM-Wartezeit weiter.

13.2.2 Daten auswerten

Vor der Auswertung der Daten muss die Ausgabedatei des KDCMON-Laufes zunächst mit dem Tool KDCPMSM von PAM- bzw. BTAM-Format in SAM-Format umgewandelt werden.
Danach müssen die Sätze mit dem BS2000-Dienstprogramm SORT nach dem Zeitstempel, der in jedem Satz vorhanden ist, sortiert werden. Dies ist notwendig, da nicht garantiert ist, dass die Sätze zeitlich sortiert in der Datei stehen.

13.2.2.1 Daten in SAM-Format umwandeln und sortieren

Zum Umwandeln und Sortieren wird dem Anwender die Prozedur PAMSAM in der Prozedur-Bibliothek

SYSPRC.UTM.070 ausgeliefert.

Vor dem Prozeduraufruf müssen Sie zunächst die KDCMON-Datei zuweisen:

- Plattendateien

```
/SET-FILE-LINK LINK-NAME=KDCMON,FILE-NAME=kdcmonfile,ACCESS-METHOD=UPAM
```

- Banddateien

```
/SET-FILE-LINK LINK-NAME=KDCMON,FILE-NAME=kdcmonfile,ACCESS-METHOD=BTAM
/MODIFY-FILE-ATTRIBUTES FILE-NAME=kdcmonfile -
/ ,SUPPORT=*TAPE(VOLUME=xxxxxxx,DEVICE-TYPE=TAPE-C4)
```

Die Prozedur PAMSAM rufen Sie wie folgt auf:

```
/CALL-PROCEDURE NAME=SYSPRC.UTM.070(PAMSAM),PROCEDURE-PARAMETERS=( -
/                                  [,KDCPMSM=kdcpmsm-progname][,SAMFILE=samfile] -
/                                  [,SORT=sortprogram])
```

Bedeutung der Parameter und Standardwerte

kdcmonfile

PAM- bzw. BTAM-Datei mit den aufgezeichneten Daten, die umgesetzt wird.

kdcpmsm-progname

Name des Tools KDCPMSM in Datei SYSPRG.KDCMON.*nnn*.KDCPMSM:

nnn=190 für BS2000 OSD/BC V10.0

nnn=200 für BS2000 OSD/BC V11.0

samfile

Name der Ausgabedatei im SAM-Format, in die PAMSAM die nach dem Zeitstempel sortierten Datensätze schreiben soll.

sortprogram

Name des BS2000-Dienstprogramms SORT.

Während des Ablaufs der Prozedur wird die Datei `KDCMON.WORK` erzeugt, die nach dem Sortierlauf wieder gelöscht wird.

Nach dem Prozedurlauf stehen die in SAM-Format umgewandelten und nach dem Zeitstempel sortierten Daten in der Datei *samfile*, die im Parameter SAMFILE angegeben wurde.

Bei fehlerhafter Beendigung des Tools KDCPMSM wird der Auftragsschalter 3 auf ’on’ gesetzt.

13.2.2.2 Daten mit dem Tool KDCEVAL auswerten

Mit KDCEVAL können die sortierten Daten ausgewertet werden. Die Auswertung kann im Dialog und im Batch durchgeführt werden. KDCEVAL benötigt zum Ablauf die Eingabe von Steuerparametern.

In einem Lauf können nur Daten **einer** Anwendung ausgewertet werden. Der Anwender hat die Möglichkeit, durch Angabe eines gewünschten Zeitintervalls (Parameter TIME=) die Auswertung auf einen Teil der erfassten Daten zu beschränken. Wurde während des Auswertungsintervalls für die Anwendung mehrfach die Datenerfassung ein- und ausgeschaltet, so wird für jeden Erfassungszeitraum (von KDCMON=ON bis KDCMON=OFF) eine getrennte Auswertung durchgeführt, wobei maximal 10 solcher Erfassungszeiträume möglich sind.

Das Auswertungstool KDCEVAL wird gestartet mit:

```
/SET-FILE-LINK LINK-NAME=KDCMON,FILE-NAME=samfile              -
/                              ,BUFFER-LENGTH=BY-CATALOG       -
/                              ,BLOCK-CONTROL-INFO=BY-CATALOG
/START-KDCEVAL
```

Bedeutung des Parameters

samfile

Name der Ausgabedatei von PAMSAM, d.h. die SAM-Datei mit den sortierten Datensätzen.

Nach dem Start des Auswertungsprogramms im Dialog fordert KDCEVAL mit der folgenden Meldung die Eingabe von Steuerparametern an:

```
KDCEVAL : PLEASE ENTER COMMANDS OR "HELP" OR "END"
```

Steuerparameter von KDCEVAL

Das Programm liest die Parameter von SYSDTA. Die einzelnen Kommandos, mit denen Sie die Auswertung steuern können, haben folgendes Format:

APPLINAME appliname

Name der Anwendung, für die die Auswertung durchgeführt werden soll.
Befinden sich in der Datei Daten mehrerer Anwendungen, so muss für jede Anwendung eine separate Auswertung durchgeführt werden. Eine Auswertung für mehrere Anwendungen in einem KDCEVAL-Lauf ist nicht möglich.

TIME FROM={ t1 | START }, TO={ t2 | STOP }

Zeitangabe zur Definition des Auswertungsintervalls.

FROM=t1

Startzeitpunkt der Auswertung in Sekunden.
Die Zeitangabe erfolgt relativ zum Start von KDCMON.

FROM=START

Es wird vom Anfang der Datei an ausgewertet.

TO=t2

Endzeitpunkt der Auswertung in Sekunden.
Die Zeitangabe erfolgt relativ zum Start von KDCMON.

TO=STOP

Es wird bis zum Dateiende ausgewertet.
Für *t1* und *t2* gilt:
Minimalwert: 0
Maximalwert: 99999999

LIST { ($liste_1$,[$liste_2$,...,$liste_{n]}$ [,TABLE]) | (STD [,TABLE]) | (ALL [,TABLE]) }

$liste_1$, $liste_2$,...,$liste_n$

Namen der Einzellisten, die aufbereitet werden sollen. Folgende Listen sind möglich: TASKS, SUMM, TIMES, KCOP, WAIT, TCLASS, TACCL, TACPT, TACLIST, TRACE oder TRACE2, näheres zu diesen Listen finden Sie im Abschnitt "Auswertungslisten". Die Listen TRACE und TRACE2 dürfen nicht zusammen angegeben werden.

STD

Diese Auswertung umfasst die Listen TASKS, SUMM, TIMES und TCLASS.

ALL

Die Auswertung umfasst alle Listen außer TRACE und TRACE2.
Sollen zusätzlich TRACE oder TRACE2 ausgewertet werden, dann geben Sie LIST (ALL, TRACE) bzw. LIST (ALL, TRACE2) an.

TABLE

Wird zusätzlich TABLE angegeben, dann werden die Listen in einem Tabellenformat erzeugt, das auf einem PC mit Excel oder einem anderen Tabellenkalkulator weiterverarbeitet werden kann, siehe "Auswertungsdaten auf dem PC bearbeiten". TABLE wirkt nur auf die Einzellisten TASKS, TIMES, TCLASS, TACCL, TACPT und TACLIST.

OPTION **D**ECIMAL-**S**EPARATOR={ **C**OMMA | **P**OINT }, SHOW-TSN={ **F**IRST | **A**LL }

DECIMAL-SEPARATOR=COMMA

Als Dezimaltrennzeichen wird das Komma verwendet.

DECIMAL-SEPARATOR=POINT

Als Dezimaltrennzeichen wird der Punkt verwendet, Standardwert.

SHOW-TSN=FIRST

In der Liste TRACE2 wird in aufeinanderfolgenden Sätzen mit identischer TSN die TSN nur im ersten Satz dieser Folge ausgegeben. Die Folgesätze einer solchen Folge von Sätzen mit gleicher TSN enthalten im TSN-Feld Anführungszeichen (").

Dies ist der Standardwert.

SHOW-TSN=ALL

In der Liste TRACE2 wird die TSN in jedem Satz ausgegeben. Dies kann dann sinnvoll sein, wenn die Ausgabeliste mit einem Programm bearbeitet oder ausgewertet werden soll.

END

Mit diesem Kommando wird die Parametereingabe beendet.

Bei Auswertungen im Dialog kann auch das Kommando HELP eingegeben werden. Es werden dann die Syntax der Kommandos und die möglichen Listennamen ausgegeben.

Fehler und Meldungen

- Bei fehlerhafter Beendigung von KDCEVAL wird der Auftragsschalter 3 auf ’on’ gesetzt.
- Fehlt eines der Kommandos APPLINAME, TIME oder LIST, so wird die Auswertung mit folgender Fehlermeldung abgebrochen:

  ```
  MANDATORY COMMAND MISSING
  ```
- Bei einem Syntaxfehler erscheint die folgende Meldung und das fehlerhafte Kommando wird angezeigt:

  ```
  ERROR IN COMMAND
  ```
- Sind die Zeitangaben *t1* und *t2* inkonsistent, so wird folgende Meldung ausgegeben:

  ```
  KDCEVAL: WRONG TIME INPUT
  ```
- Werden in der Datei keine Sätze für die Anwendung gefunden oder sind für das Auswertungsintervall keine Daten vorhanden, so wird eine der folgenden Meldungen ausgegeben:

  ```
  NO EVALUATION : NO RECORD WITH APPLINAME FOUND
  ```

  oder

  ```
  NO EVALUATION : NO RECORD IN TIME_INTERVAL
  ```
- Bei einem DMS-Fehler werden folgende Meldungen ausgegeben:

  ```
  KDCEVAL: DMS-ERROR Dxxx FOR INPUT FILE
  KDCEVAL: NO EVALUATION
  ```

  (Dxxx = DMS-Errorcode)

- Versionsprüfung:

  Die Auswertung der KDCMON-Daten durch KDCEVAL ist nur möglich, wenn KDCEVAL die gleiche openUTM-Version wie der UTM-Systemcode hat. KDCEVAL prüft die Version der KDCMON-Daten. Erkennt KDCEVAL eine unzulässige Version, bricht KDCEVAL die Auswertung mit der folgenden Meldung ab:

  ```
  NO EVALUATION: INPUT FILE FROM INVALID UTM VERSION
  ```

Ergebnis der Auswertung mit KDCEVAL

Das Ergebnis der Auswertung schreibt KDCEVAL in die Dateien einer Dateigenerationsgruppe (FGG) mit dem Namen:

`KDCMON.appliname`

Die FGG enthält maximal 10 Generationen. Beim Start der Auswertung wird eine bereits vorhandene Dateigeneration dieses Namens gelöscht.

Wenn während eines KDCMON-Laufs die Datenerfassung mehrmals ein- und ausgeschaltet wurde, schreibt KDCEVAL die Auswertungsdaten jedes Intervalls in eine eigene Datei der FGG.

13.2.3 Auswertungsdaten auf dem PC bearbeiten

Wenn Sie bei KDCEVAL im Steuerparameter LIST zusätzlich zu den Listennamen den Operanden TABLE angeben, dann werden diese Listen in Tabellenform erzeugt. Diese Art der Aufbereitung ist nur für Listen TASKS, TIMES, TCLASS, TACCL, TACPT, und TACLIST möglich.

Diese so erzeugten Listen können Sie auf einem PC mit einem Tabellenkalkulator wie z.B. Microsoft Excel bearbeiten und grafisch aufbereiten. Für Excel wird dazu auf dem WinAdmin-Datenträger das Makro `kdceval.xls` ausgeliefert.

Dazu gehen Sie wie folgt vor:

1. Übertragen Sie die von KDCEVAL erzeugte Listen-Datei mit ftp oder openFT auf einen PC und kopieren Sie auf diesen PC das Makro `kdceval.xls` vom WinAdmin-Datenträger.

   Das Makro verlangt, dass die auszuwertende Datei die Endung `.txt` besitzt.
2. Rufen Sie das Makro `kdceval.xls` auf und lesen Sie die Listen-Datei in Excel ein. Excel erzeugt dann für jede Liste ein eigenes Tabellenblatt sowie ein Zusatzblatt mit Übersichtsinformationen. (Summary-Blatt).
3. Bearbeiten Sie die einzelnen Listen nach Ihren Wünschen, indem Sie z.B. eine Liste sortieren und anschließend in ein Kurven- oder Balkendiagramm umwandeln.

13.2.4 Auswertungslisten

Jede Auswertungsliste besteht aus

- einer Überschrift, die den Namen der Auswertungsliste enthält
- einem Kopf, der für alle Listen identisch ist
- der spezifischen Auswertungsliste

Der Listenkopf hat folgenden Aufbau:

NAME OF APPLICATION : applname DATE : yyyy-mm-dd

BS2000 VERSION :V190 openUTM VERSION: V07.0A
COMMENCEMENT TIME : rel-sec SEC. KDCMON START : hh:mm:ss

KDCEVAL VERSION: V7.0A00
END TIME : rel-sec SEC. APPLICATION RECORDING START: hh:mm:ss
APPLICATION RECORDING END : hh:mm:ss
SYSTEM INFORMATION : system info ,Number CPUs : nn, Bitmode : nn Bit
APPLICATION INFO : Appli-Mode S Cache size,loc BLKSIZE : 4K Test-Mode OFF Data Compression OFF

Die Felder haben folgende Bedeutung (soweit nicht selbsterklärend):

NAME OF APPLICATION	Name der Anwendung
DATE	Datum der Datenerfassung mit KDCMON
BS2000 VERSION	BS2000-Version der Anlage, auf der KDCMON gelaufen ist
COMMENCEMENT TIME	Anfangszeitpunkt des ausgewählten Auswertungszeitraums (relativ zum Startzeitpunkt von KDCMON)
KDCMON START	Startzeit von KDCMON
END TIME	Endzeitpunkt des ausgewählten Auswertungszeitraums (relativ zum Startzeitpunkt von KDCMON)
APPLICATION RECORDING START	Startzeit der Datenerfassung für die UTM-Anwendung
APPLICATION RECORDING END	Endzeitpunkt der Datenerfassung für die UTM-Anwendung
Cache size, loc	Größe und Speicherort des Cache. Für den Speicherort sind folgende Ausgaben möglich: PS falls der Cache im Programmraum angelegt ist. DS falls der Cache in einem Datenraum angelegt ist.

Es sind folgende Einzel-Auswertungen und Kombinationen dieser Auswertungen möglich:

TASKS	UTILIZATION OF THE UTM TASKS
SUMM	TRANSACTION EVALUATION
TIMES	DISTRIBUTION OF PROCESSING TIMES
KCOP	KDCS CALLS STATISTIC
WAIT	WAITING TIMES
TCLASS	EVALUATION OF THE TAC CLASSES
TACCL	TAC SPECIFIC TAC CLASS EVALUATION
TACPT	TAC SPECIFIC DISTRIBUTION OF PROCESSING TIMES
TACLIST	TAC SPECIFIC STATISTICS
TRACE	TASK SPECIFIC TRACES
TRACE2	TASK PERFORMANCE TRACES

Im Folgenden werden die einzelnen Auswertungslisten beschrieben.

13.2.4.1 TASKS: UTILIZATION OF THE UTM TASKS

In dieser Liste wird ein Überblick über die Auslastung der Prozesse der Anwendung gegeben. Außerdem werden der CPU-Verbrauch und die Anzahl der Ein- und Ausgabeoperationen (I/O's) für jeden einzelnen UTM-Prozess aufsummiert und für alle Prozesse der Anwendung angezeigt.

```
                          1 = Program
                          2 = System code
                          3 = Database
                          4 = Bourse Wait
|------------------------|
| TSN  | START TIME      | TASK UTILIZATION    , Number Used Tasks:   6 , Number System
Tasks:   0
| 3FYA | 16:46:32.928063 | <--------1-------><----2----><----3---><--------------4-------------->
| 3FYB | 16:46:32.917157 | <--------1------><---2---><---3---><----------------4---------------->
| 3FYC | 16:46:37.661807 | 2<-----------------------------------4--------------------------------->
| 3FYD | 16:46:32.924186 | <-----------------------------------4--------------------------------->
| 3FYE | 16:46:32.936194 | <--------1------><---2---><----3---><----------------4---------------->
| 3FYF | 16:46:32.914551 | <--------1------><---2---><----3----><----------------4---------------->
|------|-----------------|
TSN      CPU-time  Number I/O     Program      System    Database  Bourse Wait  System Task
3FYA        20056        7150       70596       38931       35553       148124            N
3FYB        19133        6904       64621       35138       32592       160865            N
3FYC         1852         933          19        5009         111       283332            N
3FYD         1084         490          46        2808          29       290326            N
3FYE        19389        6964       66356       34615       35336       156890            N
3FYF        19188        6995       65410       34683       32640       160486            N
Summ        80702       29436
```

In der Liste bedeuten:

TSN	TASK SEQUENCE NUMBER des UTM-Prozesses.
START TIME	Zeit des ersten Datensatzes dieses Prozesses (absolut).
Program	Zeitanteil des Anwendungsprogramms im UTM-Prozess. In dieser Zeit ist auch die SVC-Behandlung der SVCs enthalten, die aus den Anwendungsprogrammen aufgerufen werden.
System code	Zeitanteil des UTM-Systemcodes.
Database	Zeitanteil, der zur Bearbeitung der Datenbankaufrufe benötigt wurde. Erfordert die Bearbeitung eines Datenbankaufrufs einen Prozesswechsel, so beinhaltet diese Zeit auch die Wartezeit des UTM-Prozesses.
Bourse Wait	Zeitanteil, den der Prozess auf neue Aufträge an der Auftragswarteschlange gewartet hat.
System Task	Gibt an, ob diese Task ein UTM-System-Prozess der Anwendung ist.

Bei den in den Spalten Program, System, Database und Bourse Wait ausgegebenen Zeiten handelt es sich um Real-Zeiten. Die Einheit ist in Millisekunden (genauso wie für die CPU-Zeit). Für die Auswertung TASKS ist ein Herabsetzen der Prozess-Anzahl während des Auswertungsintervalls zu vermeiden, denn es führt zu verzerrten Ergebnissen. Für einen solchen Anwendungsfall sollten Sie getrennte Auswertungsintervalle verwenden.

13.2.4.2 SUMM: TRANSACTION EVALUATION

In dieser Liste wird ein Überblick über die Vorgänge und Transaktionen für den Auswertungszeitraum gegeben. Es werden nur solche Transaktionen berücksichtigt, die vollständig im Auswertungszeitraum liegen. Zusätzlich gibt das Auswertungstool KDCEVAL den CPU-Verbrauch von allen Teilprogrammläufen aus, die im Auswertungsintervall beendet wurden. Die Liste hat folgendes Format:

```
|-------------------------------------------------------------|
| COUNT OF TRANSACTIONS                        :       8229 |
|                                                           |
| COUNT OF SERVICES                            :       8229 |  1)
|                                                           |
| COUNT OF DIALOG STEPS                        :       8229 |
|                                                           |
| NUMBER OF DIALOG STEPS PER SECOND            :      28.08 |
|                                                           |
| NUMBER OF DB CALLS PER TRANSACTION           :          9 |
|                                                           |
| TOTAL CPU-TIME USED IN MSEC                  :      77741 |  2)
|-------------------------------------------------------------|
```

1) SERVICE = Vorgang

2) In dieser Zeile wird die Summe des CPU-Verbrauchs der Teilprogrammläufe ausgegeben. Darin ist auch der CPU-Verbrauch im UTM- und Betriebssystemcode enthalten, soweit er innerhalb der Teilprogrammläufe anfällt, sowie die Start- und Endeverarbeitung für die Teilprogrammläufe in openUTM. Andere Aktionen der UTM-Prozesse, die nicht direkt zu Teilprogrammläufen gehören, sind nicht in diesen Werten enthalten.

13.2.4.3 TIMES: DISTRIBUTION OF PROCESSING TIMES

In dieser Liste wird in tabellarischer Form eine Verteilung der Bearbeitungszeiten in Millisekunden für die Teilprogramme ausgegeben. In diesen Zeiten ist die Wartezeit vor der Bearbeitung durch openUTM nicht enthalten.

Die Liste hat folgendes Format:

PROCESSING TIMES (MSEC)	NUMBER	PERCENT
0 - 100	76	58.91
101 - 200	45	34.88
201 - 500	7	5.42
501 - 1000	1	0.77
1001 - 2000	0	0.00
2001 - 5000	0	0.00
5001 - 10000	0	0.00
10001 - 20000	0	0.00
20001 - 50000	0	0.00
50001 - 100000	0	0.00
> 100000	0	0.00

In dieser Liste ist die Anzahl der vollständigen Teilprogrammläufe und der prozentuale Anteil für die jeweilige Zeitklasse angegeben.

13.2.4.4 KCOP: UTM CALLS STATISTIC

In dieser Tabelle ist angegeben, wie oft die einzelnen UTM-Aufrufe im Auswertungszeitraum aufgetreten sind.

Unter *others* sind Aufrufe aufgeführt, die nicht in der Liste der in KCEVAL bekannten Aufrufe enthalten sind.

In dieser Liste sind auch Aufrufe enthalten, die openUTM für interne Bearbeitungen aufruft und die dem Anwender nicht zur Verfügung stehen:

CONT	Aufruf nach einer Formatierung oder einer Datenbankkommunikation
ADMI	UTM-Administrationsaktion
WAIT	Ende der Bearbeitung eines Programmlaufes
NOOP	Messdaten für DB-Aufrufe schreiben

Die KCOP-Liste hat folgendes Format:

```
|-----------------------------|    |-----------------------------|
| OP   | OM   |        NUMBER   |    | OP   | OM   |        NUMBER   |
|-----------------------------|    |-----------------------------|
| ADMI                    7   |    | MPUT  HM                0   |
| APRO  AM                0   |    | MPUT  ID                0   |
| APRO  DM                0   |    | MPUT  NE                0   |
| APRO  IN                0   |    | MPUT  NT            34761   |
| CONT                17338   |    | MPUT  PM                0   |
| CTRL  AB                0   |    | MPUT  RM                0   |
| CTRL  EC                0   |    | NOOP                    0   |
| CTRL  PE                0   |    | PADM  AC                0   |
| CTRL  PR                0   |    | PADM  AI                0   |
| CTRL  SC                0   |    | PADM  AT                0   |
| DADM  CS                0   |    | PADM  CA                0   |
| DADM  DA                0   |    | PADM  CS                0   |
| DADM  DL                0   |    | PADM  OK                0   |
| DADM  MA                0   |    | PADM  PI                0   |
| DADM  MV                0   |    | PADM  PR                0   |
| DADM  RQ                0   |    | PEND  ER                0   |
| DADM  UI                0   |    | PEND  FC                0   |
| DGET  BF                0   |    | PEND  FI             8231   |
| DGET  BN                0   |    | PEND  FR                0   |
| DGET  FT                0   |    | PEND  KP                0   |
| DGET  NT                0   |    | PEND  PA                0   |
| DGET  PF                0   |    | PEND  PR                0   |
| DGET  PN                0   |    | PEND  PS                0   |
| DPUT  NE                0   |    | PEND  RE                0   |
| DPUT  NI                0   |    | PEND  RS                0   |
```

```
| DPUT  NT                0   |    | PEND  SP                0   |
| DPUT  RP                0   |    | PGWT  CM                0   |
| DPUT  QE                0   |    | PGWT  KP                0   |
| DPUT  QI                0   |    | PGWT  PR                0   |
| DPUT  QT                0   |    | PGWT  RB                0   |
| DPUT  +I                0   |    | PGWT  RT                0   |
| DPUT  -I                0   |    | PGWT  ST                0   |
| DPUT  +T                0   |    | PTDA                    0   |
| DPUT  -T                0   |    | QCRE  NN                0   |
| FGET                    0   |    | QCRE  WN                0   |
| FPUT  NE                0   |    | QREL  RL                0   |
| FPUT  NT                0   |    | RSET                 6907   |
| FPUT  RP                0   |    | SGET  GB                0   |
| FPUT  UF                0   |    | SGET  KP                0   |
| GTDA                    0   |    | SGET  RL                0   |
| INFO  CD                0   |    | SGET  US                0   |
| INFO  CK                0   |    | SIGN  CK                0   |
| INFO  DT                0   |    | SIGN  CL                0   |
| INFO  FH                0   |    | SIGN  CP                0   |
| INFO  GN                0   |    | SIGN  OB                0   |
| INFO  LO                0   |    | SIGN  OF                0   |
| INFO  PC                0   |    | SIGN  ON                0   |
| INFO  SI                0   |    | SIGN  ST                0   |
| INIT                 8230   |    | SMSG                    0   |
| INIT  PU                0   |    | SPUT  DL                0   |
| INIT  MD                0   |    | SPUT  ES                0   |
| LPUT                    0   |    | SPUT  GB                0   |
| MCOM  BC                0   |    | SPUT  MS                0   |
| MCOM  EC                0   |    | SPUT  US                0   |
| MGET                 8230   |    | SREL  GB                0   |
| MGET  NT                0   |    | SREL  LB                0   |
| MPUT  CM                0   |    | UNLK  DA                0   |
| MPUT  EM                0   |    | UNLK  GB                0   |
| MPUT  ES                0   |    | UNLK  US                0   |
| MPUT  GC                0   |    | WAIT                 8232   |
| OTHERS                  0   |    |                             |
|-----------------------------|    |-----------------------------|
```

13.2.4.5 WAIT: WAITING TIMES

Zur Ermittlung von Stausituationen fügt openUTM bei eingeschaltetem KDCMON in regelmäßigen Zeitabständen Messaufträge in die Auftragswarteschlange ein. Mit Hilfe des Zeitpunktes, zu dem der Auftrag eingefügt wurde und dem Bearbeitungszeitpunkt kann die Wartezeit der Aufträge in der UTM-Warteschlange ermittelt werden. Der Zeitabstand zwischen den einzelnen Pseudoaufträgen beträgt etwa 10 Sekunden.

In der WAIT-Liste wird Folgendes protokolliert:

- In der Spalte WAITING TIME wird die ermittelte Wartezeit für jeden Pseudoauftrag in Sekunden ausgegeben.
- Für diese Wartezeiten berechnet das Auswertungstool KDCEVAL zusätzlich den Maximal-, Minimal- und Mittelwert in Sekunden und zeigt diese Werte unter UTM WAITING TIMES an.
- In der Spalte NUMBER OF TASKS wird die Anzahl der Prozesse, die zu diesem Zeitpunkt in der Anwendung zur Verfügung standen, ausgegeben. Die UTM-System-Prozesse sind in dieser Zahl nicht berücksichtigt.
- Die BCAM-Wartezeiten werden unter BCAM WAITING TIMES in Sekunden ausgegeben. BCAM misst die Zeit, die Eingabe-Nachrichten für die Anwendung im Transportsystem BCAM verbringen, bevor openUTM sie abholt.

  Bei NUMBER OF MESSAGES wird die Anzahl der Nachricht angezeigt, für die BCAM eine Wartezeit geliefert hat. Beim Einschalten der KDCMON-Aufzeichnung (und damit auch der Zeiterfassung bei BCAM) sind i.a. bereits Nachrichten im Nachrichten-Pool von BCAM. Wenn openUTM diese Nachrichten bei BCAM abholt, erhält openUTM für sie keine Wartezeit. Diese Nachrichten werden im Wert NUMBER OF MESSAGES nicht mitgezählt.

Wenn die Wartezeiten zu lang werden, sollte die Anzahl der UTM-Prozesse erhöht werden.

Die WAIT-Liste hat folgendes Format:

TIME STAMP	WAITING TIME	NUMBER OF TASKS
10:33:35.742	0.018	1
10:33:45.781	0.008	2
10:33:55.841	0.007	3
10:33:66.027	0.008	3
10:33:76:087	0.008	3
10:33:86:112	0.007	3
10:33:96:123	0.007	3

```
                      UTM WAITING TIMES:
TIME STAMP :  10:33:35.742    WAITING TIME MAXIMUM :        0.018
TIME STAMP :  10:33:41.740    WAITING TIME MINIMUM :        0.007
NUMBER OF ENTRIES:       7    WAITING TIME AVERAGE :        0.009
                     BCAM WAITING TIMES:
TIME STAMP :  10:23:35.742    WAITING TIME MAXIMUM :        0.728
TIME STAMP :  10:39:35.740    WAITING TIME MINIMUM :        0.027
NUMBER OF MESSAGES:    200    WAITING TIME AVERAGE :        0.125
```

13.2.4.6 TCLASS: EVALUATION OF THE TAC CLASSES

Die TCLASS-Liste enthält in tabellarischer Form einen Überblick über die Auftragsbearbeitung von TACs der einzelnen TAC-Klassen (1 bis 16). In der Auswertung sind in der TAC-Klasse 0 alle die Dialog-TACs zusammengefasst, denen bei der UTM-Generierung mit KDCDEF keine TAC-Klasse zugeordnet wurde.

Bei der UTM-Generierung kann der Anwender festlegen, wieviele Prozesse zu einem Zeitpunkt maximal für eine TAC-Klasse arbeiten dürfen. Ist diese Anzahl erreicht, so werden weitere Aufträge in eine TAC-Klassen-spezifische Warteschlange eingereiht.

TAC-CLASS	NUMBER CALLS	DISTRIBUTION IN PERCENT			AVERAGE WAIT TIME (IN MSEC)	MAXIMUM WAIT TIME (IN MSEC)	MINIMUM WAIT TIME (IN MSEC)
		NUMBER CALLS	WAIT TIME=0	WAIT TIME>0			
0	21	30.87					
1	2	2.94	100.00	0.00	0	0	0
2	4	5.88	75.00	25.00	3000	5000	1000
3	3	4.41	100.00	0.00	0	0	0
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
8	5	7.35	80.00	20.00	5000	8000	2000
9	2	2.94	90.00	10.00	4000	0	0
10	3	4.41	100.00	0.00	0	5000	1000
11	4	5.88	75.00	25.00	3000	0	0
12	3	4.41	100.00	0.00	0	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	8000	2000
16	5	7.35	80.00	20.00	5000		
			90.00	10.00	4000		

41 DIALOG TACS WERE CALLED

27 ASYNCHRONOUS TACS WERE CALLED

Die TCLASS-Liste enthält folgende Angaben:

- In der Spalte NUMBER CALLS wird für eine TAC-Klasse die Anzahl der TAC-Aufrufe im Auswertungszeitraum angegeben.
- Die Spalte DISTRIBUTION IN PERCENT enthält Prozentwerte.
  Die Unterspalte NUMBER CALLS gibt den prozentualen Anteil der Aufrufe einer TAC-Klasse zur Anzahl aller TAC-Aufrufe an. Die nächsten beiden Spalten enthalten eine prozentuale Aufteilung der Aufrufe dieser TAC-Klasse in
  - Aufrufe, die sofort bearbeitet wurden (WAITTIME=0) und
  - Aufrufe, die in eine TAC-Klassen-spezifische Warteschlange eingereiht wurden (WAITTIME>0)
- Die Werte in den Spalten AVERAGE / MINIMUM / MAXIMUM WAIT TIME beziehen sich auf die Aufträge, die openUTM vorübergehend in eine TAC-Klassen-spezifische Warteschlange verdrängt hat. Es wird die mittlere, minimale bzw. maximale Wartezeit eines Auftrags pro TAC-Klasse angezeigt.

> **i** Die mittlere Wartezeit von Aufträgen pro TAC-Klasse kann im laufenden Betrieb einer Anwendung auch mit dem Administrationskommando KDCINF TACCLASS oder mit der entsprechenden Funktion bei WinAdmin/WebAdmin oder KDCADMI abgefragt werden.

Wartezeit bei Dialog-Aufträgen

Bei Dialog-Aufträgen ist die Wartezeit die Zeitspanne zwischen der Entgegennahme des Auftrags durch die Anwendung (Abholen des Auftrags von der Warteschlange der Anwendung) und dem Start des Teilprogramms.

Wartezeit bei Asynchron-Aufträgen

openUTM erfasst auch die Wartezeit von Asynchron-Aufträgen, die sich wie folgt definiert:

Asynchron-Auftrag	**Definition „Wartezeit“**
Eingabe Asynchron-TAC	Zeitspanne zwischen der Entgegennahme des Auftrags durch openUTM und dem Starten des Asynchron-Vorgangs
FPUT-Aufruf im Teilprogramm	Zeitspanne zwischen dem Ende der Transaktion, in der der FPUT-Auftrag ausgeführt wurde, und dem Starten des Asynchron-Vorgangs
DPUT-Aufruf im Teilprogramm	Zeitspanne zwischen der Umwandlung des DPUT in einen FPUT und dem Starten des Asynchron-Vorgangs

Wurde der Asynchron-Auftrag nicht im aktuellen Anwendungslauf erzeugt, so wird als asynchrone Wartezeit immer die Zeitdifferenz zwischen dem Start der Anwendung und dem Start des Asynchron-Auftrags genommen.

13.2.4.7 TACCL: TAC SPECIFIC TAC CLASS EVALUATION

In der Liste TACCL sind die gleichen Informationen wie in der TCLASS-Liste aufgeführt, aufgegliedert auf die einzelnen Transaktionscodes. Es sind alle TACs aufgeführt, die im Auswertungszeitraum aufgerufen wurden. Die TACs sind in der Reihenfolge des ersten Auftretens aufgeführt. Die Bedeutung der einzelnen Spalten kann der Beschreibung des TCLASS-Listenformates entnommen werden.

TAC	TAC-CLASS	NUMBER CALLS	DISTRIBUTION IN PERCENT			AVERAGE WAIT TIME (IN MSEC)
			NUMBER CALLS	WAIT TIME=0	WAIT TIME> 0	
TAC1	0	10	22.50			
TAC2	5	5	11.25	60.00	40.00	1000
TAC3	2	1	2.50	0.00	100.00	3256
TAC4	13	12	23.40	40.00	60.00	2000
TAC5	15	17	26.20	55.00	45.00	4000
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

Für TACs der TAC-Klasse 0 werden keine Angaben für WAIT TIME eingetragen.

13.2.4.8 TACPT: TAC SPECIFIC DISTRIBUTION OF PROCESSING TIMES

In dieser Tabelle sind für alle im Auswertungszeitraum bearbeiteten TACs die minimale (MIN), die maximale (MAX) und die mittlere (MEAN) Bearbeitungszeit in Millisekunden aufgeführt. Es werden nur solche TACs aufgenommen, deren Start- und Beendigungszeitpunkt innerhalb des Auswertungszeitraums liegen. Die Liste hat folgendes Format:

TAC	PROCESSING TIME PER TAC (IN MSEC)		
	MEAN	MIN	MAX
TAC1	150000	125000	175000
TAC2	35000	19000	45000
TAC3	24500	20000	29000

Die Tabelle ist nach mittleren Bearbeitungszeiten fallend sortiert. Angezeigt werden nur TACs mit einer mittleren Bearbeitungszeit > 0. Bei den ausgegebenen Zeiten handelt es sich um Realzeiten (elapsed time).

13.2.4.9 TACLIST: TAC SPECIFIC STATISTICS

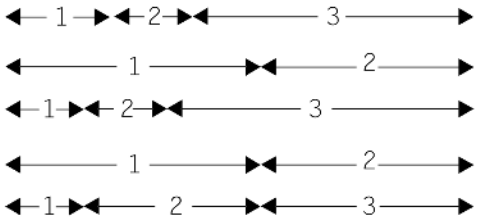
In dieser Liste werden die folgenden TAC-spezifischen Informationen erfasst:

- die durchschnittliche KB-Größe (Spalte AVERAGE SIZE OF KB)
- die durchschnittliche Anzahl der Datenbankaufrufe, getrennt nach
  - Aufrufen aus den Anwendungsteilprogrammen (Spalte FROM USER) und
  - Aufrufen, die zur Koordination zwischen openUTM und dem Datenbanksystem dienen (Spalte FROM SYSTEM)
- die Aufteilung der Bearbeitungszeit in

  1: program
  2: system code
  3: data base (die Bedeutung dieser Zeiten ist die gleiche wie in der TASKS-Liste)

Die Liste hat folgendes Format:

TAC	NUMBER CALLS	AVERAGE DB CALLS		AVERAGE SIZE OF KB
		FROM USER	FROM SYSTEM	
TAC1	5	15	2	256
TAC2	18	0	0	1024
TAC3	3	200	2	3000
TAC4	70	0	0	1024
TAC5	500	32	2	30000
.	.	.	.	.
.	.	.	.	.



Die Liste ist nicht sortiert, die TACs sind in der Reihenfolge aufgelistet, in der sie zuerst auftreten.

Es werden nur die TACs berücksichtigt, deren Start- und Beendigungszeitpunkt innerhalb des Auswertungsintervalls liegen.

13.2.4.10 TRACE: TASK SPECIFIC TRACES

Zur genaueren Analyse des Ablaufs einer UTM-Anwendung können TRACE-Listen erstellt werden. In diesen Listen werden in zeitlicher Reihenfolge für die einzelnen UTM-Prozesse alle UTM-Aufrufe aufgelistet.

In einer Liste können aus Platzgründen die Daten von maximal 6 Prozessen enthalten sein. Wenn für den Auswertungszeitraum Daten für mehr als 6 Prozesse existieren, so erzeugt das Auswertungstool KDCEVAL eine zweite TRACE-Liste mit den nächsten maximal 6 Prozessen und hängt sie in der gleichen Datei an die erste Liste an. Dadurch ist eine Analyse des zeitlichen Ablaufs von maximal 12 Prozessen möglich.

Sind in dem Auswertungszeitraum Daten von mehr als 12 Prozessen vorhanden, so erstellt KDCEVAL die Auswertungslisten für 12 Prozesse. Für die restlichen Prozesse wertet KDCEVAL keine Daten aus und protokolliert dies durch eine Meldung mit Angabe der TSNs.

Die Liste ist in zeitlicher Reihenfolge sortiert.

Die Spalte TIME STAMP enthält den Zeitstempel des entsprechenden Aufrufs, der protokolliert wurde (in Mikrosekunden).

Die Liste TRACE erfasst folgende Ereignisse und Daten:

- den aufgerufenen Transaktionscode (TAC)
- die Transaktions-ID. In openUTM wird für jede Transaktion eine eindeutige Transaktions-ID vergeben. Dieses Kennzeichen wird auch den angeschlossenen Datenbanken an der UTM-DB-Schnittstelle übergeben. Damit wird es u.a. möglich, Traces der Datenbank mit diesen Traces von openUTM zu koppeln und Zusammenhänge zwischen UTM- und DB-Abläufen herzustellen. Die Transaktions-ID besteht aus vier Teilen:

  SC Session Counter: er zählt die Anwendungsläufe. Nach Neugenerierung ist er 1, bei jedem Start der Anwendung wird er um 1 erhöht.

  VC Vorgangs Counter: er zählt die Vorgänge innerhalb eines Anwendungslaufes und läuft bis 16 777 216 (2^{24}).

  TC Transaction Counter: er zählt die Transaktionen innerhalb eines Vorgangs und läuft bis 32 768 (2^{15}).

  VN Vorgangs Nummer: das ist die Nummer einer UTM-internen Tabelle zur Verwaltung der Vorgänge.

  Diese vier Teile werden nach dem KDCS-Aufruf INIT protokolliert.
  Für den Anwender sind die Angaben zu VC und TC interessant.

  - Alle UTM-Aufrufe mit ihren Operationsmodifikationen. Auch UTM-interne Aufrufe (WAIT, CONT, ...) werden aufgeführt.

    Zusätzlich wird protokolliert:

    KCMF bei Aufrufen, bei denen KCMF relevant ist

    KCRN bei Aufrufen, bei denen KCRN relevant ist

    KCLT bei den Aufrufen PADM/DADM

- bei einem Abbruch mit PEND ER/ FR als Diagnoseinformation:
  - der TAC des Teilprogramms, das den Abbruch verursacht hat
  - die Returncodes KCRCDC und KCRRCC
  - VC und TC, für die Zuordnung zum abgebrochenen Vorgang
- bei einem Aufruf PEND RS als Diagnoseinformation:
  - der TAC des aktuellen Teilprogramms
  - VC und TC, für die Zuordnung zum Vorgang
- bei DB-Aufrufen FITA und CATA:
  - VC und TC, für die Zuordnung zum Vorgang

- Alle Datenbankaufrufe, wenn die UTM-Anwendung koordiniert mit einem Nicht-XA-Datenbanksystem zusammenarbeitet (generiert mit KDCDEF-Steueranweisung DATABASE).

  Aufrufe an ein solches Datenbanksystem werden mit folgendem Eintrag protokolliert:

  opcode db-time

  Dabei ist *opcode* der Operationscode des Datenbankaufrufs gemäß IUTMDB-Schnittstelle. Sehen Sie dazu die Beschreibung der DB Diagarea im openUTM-Handbuch „Meldungen, Test und Diagnose auf BS2000-Systemen “, Kapitel UTM-Dump. *db-time* gibt an, wieviel Zeit zur Bearbeitung des Datenbankaufrufs benötigt wurde (Realzeit in Mikrosekunden)

  Beispiel: `FITA 115`

  Wenn für einen SESAM-DBH SESCOS eingeschaltet ist, werden Aufrufe an das Datenbanksystem SESAM durch folgenden Eintrag protokolliert, der die Zuordnung zwischen TRACE-Liste und SESCOS-Auswertung ermöglicht:

  ppxyttttnnnmmmmmmmm

  pp
  : Operationscode des DB-Aufrufs als Hexawert, z.B. 14=DBFITA
    Werte siehe DB Diagarea

  x
  : Konfigurationsname des SESAM-DBH
    Wertebereich: 'BLANK', A-Z, 0-9
    Wert ‘-‘: Information wurde von SESAM nicht versorgt

  y
  : Kommunikationsname des SESAM-DBH
    Wertebereich: 'BLANK', A-Z, 0-9

  tttt
  : TSN der DBH-Task

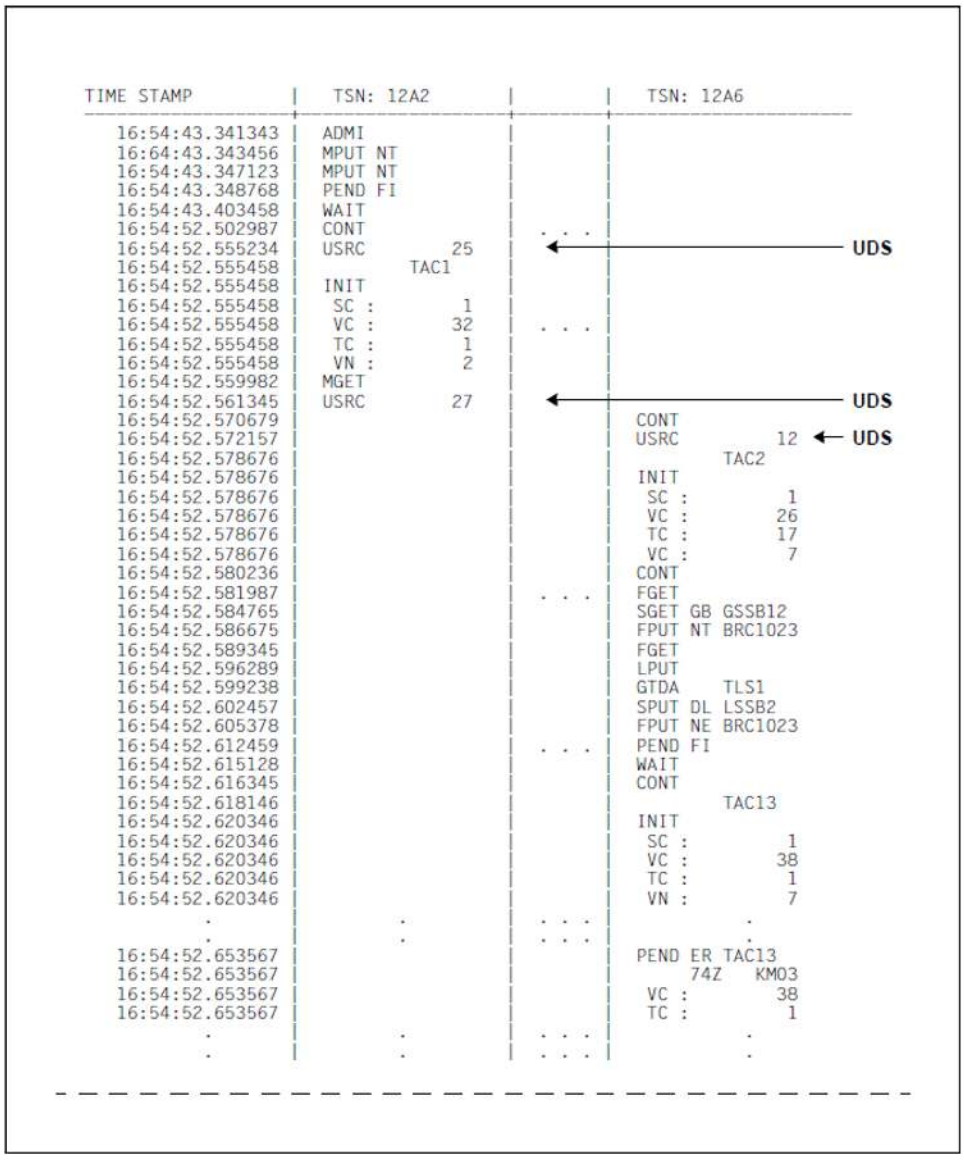
  nnn
  : laufende Nummer der SESCOS-Datei

  mmmmmmmm
  : laufende Nummer in der SESCOS-Datei.
    Die Zähler *nnn* und *mmmmmmmm* werden mit führenden Leerzeichen angegeben.

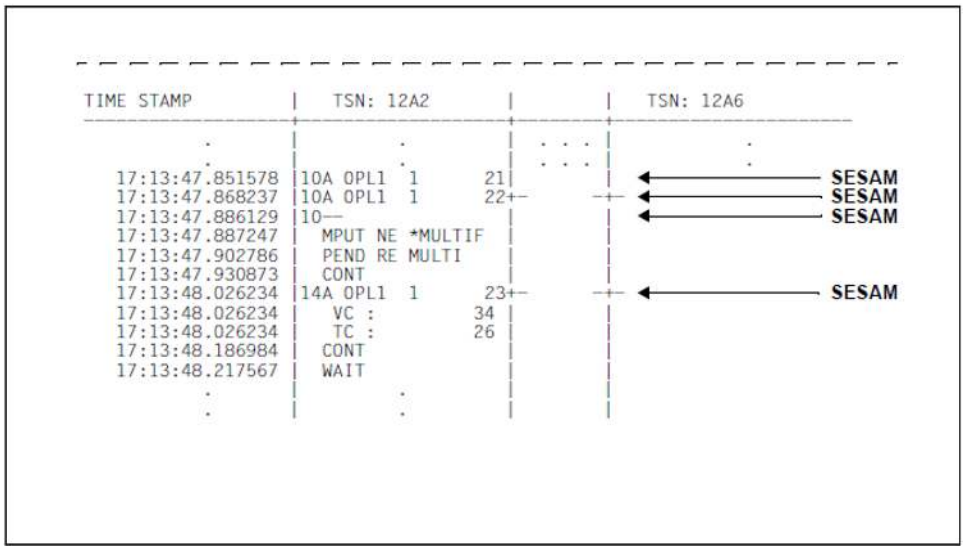
Solange kein Prozesswechsel stattfindet, sind alle Aufrufe zur Bearbeitung eines Dialogschrittes in der gleichen TSN-Spalte hintereinander aufgeführt. Nach PEND PA/PR/SP kann bei einem Wechsel der TAC-Klasse ein Prozesswechsel auftreten. Die Unterbrechung eines Prozesses durch das Betriebssystem wird dadurch sichtbar, dass in der Bearbeitung eines Dialogschrittes die Aufrufe in einer anderen Prozess-Spalte fortgesetzt werden.

Es folgt ein Beispiel für die Darstellung der TRACE-Liste. Sie zeigt einen Ausschnitt aus dem Ablauf einer UTM-Anwendung mit Datenbankaufrufen an UDS und SESAM.

Die Auswertungsliste TRACE hat folgendes Format:



TRACE-Liste (Teil 1)



TRACE-Liste (Teil 2)

Im Beispiel wurden zum besseren Verständnis die Datenbankaufrufe durch Pfeile gekennzeichnet, die in der Original TRACE-Liste nicht enthalten sind.

13.2.4.11 TRACE2: TASK PERFORMANCE TRACE

Die Auswertungsliste TRACE2 listet die wichtigsten Ereignisse in den Teilprogrammen der Anwendung sequenziell auf. Da die Auswertung nicht wie in der Liste TRACE in Spalten für die UTM-Prozesse erfolgt, kann die Liste TRACE2 beliebig viele Prozesse anzeigen. TRACE2 enthält zusätzlich zu den Einträgen der Auswertung TRACE wichtige Daten zur Performanceanalyse.

Die Einträge in der Auswertung sind in zeitlicher Reihenfolge sortiert. Die Spalte TIME STAMP enthält den Zeitstempel des Ereignisses (in Mikrosekunden).

Die Auswertungsliste TRACE2 erfasst folgende Ereignisse und Daten:

- Start eines Teilprogramms als Eintrag `strt >>> tac` mit
  - Transaktionscode des Teilprogramms
  - TAC-Klasse
  - aktueller I/O- und CPU-Stempel (in Millisekunden)
  - Wartezeit der Nachricht in BCAM
  - Wartezeit des Auftrags in der TAC-Klasse
- Alle UTM-Funktionsaufrufe mit Operationscode und -modifikation und zusätzlich folgende Informationen:
  - KCMF bei Aufrufen, bei denen KCMF relevant ist
  - KCRN bei Aufrufen, bei denen KCRN relevant ist
  - KCLT bei den Aufrufen PADM und DADM
  - Bei PEND-Aufrufen mit KCOM = ER/FR/RS und bei den Datenbankaufrufen FITA und CATA die Transaction-ID (SC,VC,TC und VN) für die Zuordnung zum abgebrochenen Vorgang.
  - Wenn KCRCCC ungleich 0, die Returncodes KCRCCC und KCRCDC und die Transaction-ID (SC,VC,TC und VN).
- Alle Datenbankaufrufe, wenn die UTM-Anwendung koordiniert mit dem DB-System zusammenarbeitet. Die Aufrufe an das Datenbanksystem werden protokolliert mit einem Eintrag DBCL *opcode db-time*.

  opcode und *db-time* haben die gleiche Bedeutung wie bei der Liste TRACE.

  Wenn auf der Seite des DBH der SESCOS eingeschaltet ist, wird bei Aufrufen an das Datenbanksystem SESAM zusätzlich eine Information
  `SESCOS-INFO -->` *xyttttnnnmmmmmmmmm* protokolliert. Die Werte ermöglichen die Zuordnung von TRACE2-Listen und SESCOS-Auswertung. Sehen Sie dazu auch die Beschreibung der Felder für die TRACE-Liste in Abschnitt "TRACE: TASK SPECIFIC TRACES".
- Ende des Teilprogramms als Eintrag `WAIT end<<<<` mit
  - CPU-Verbrauch im Teilprogramm in Mikrosekunden in Spalte „CPU“
  - Anzahl I/Os im Teilprogramm in Spalte „I/0“
  - Am Ende der Spalte "TRACE" werden in vier Zeichen die Sedezimal-Werte der beiden ersten Bytes des Börsen-Announcements protokolliert, das diesen Börsen-Zyklus gestartet hat.

    Diese Information dient dem Software Service zur besseren Analyse.
- Eine logische Prozessornummer, die KDCMON liefert.

  Jede Nummer identifiziert dabei jeweils eine CPU. Die Nummer wird in der Auswertung in der Spalte „ID“ angezeigt. Auf Multiprozessoranlagen kann damit festgestellt werden, welche UTM-Prozesse parallel laufen.

Die Strukturelemente `<<<<<<` in der Liste erleichtern die Lesbarkeit der Einträge.

Die Auswertungsliste TRACE2 hat das folgende Format.

```
TIME STAMP     | TRACE                | ID TSN |   SC       VC     TC     VN|KCRN    CPU  |KCMF    I/O  |TACCLASS Q.TIME   |BCAM W.TIME
10:48:08.353714| DBCL USRC         205| 1 8E8R |                           |            |            |                  |
10:48:08.353761| CONT 31 -------------| 1 8E85 |    1   192108      1     31|            |            |                  |
10:48:08.353900| WAIT    -------------| 1  "   |                           |            |            |                  |
10:48:08.353905+ WAIT end   <<<<< 45C5+ 1  "   |<<<<<<<<<<<<<<<<<<<<<<<<<<<|        +621 |          +2 |<<<<<<<<<<<<<<   |<<<<<<<<<<<
10:48:08.353925| DBCL USRC        2065| 1 8E8R |                           |            |            |                  |
10:48:08.356002| DBCL USRC         205| 1  "   |                           |            |            |                  |
10:48:08.356213| DBCL USRC           3| 1  "   |                           |            |            |                  |
10:48:08.356219+ DBCL USRC        3370+ 1  "   +---------------------------+------------+------------+------------------+-----------
10:48:08.357268| strt >>>RDUDS        | 1 8E85 |    1   192110      1     32|       40758 |       98786 | 0 *            0 |          0
10:48:08.357321| INIT    -------------| 1  "   |                           |            |            |                  |
10:48:08.357335| MGET    -------------| 1  "   |                           |            |            |                  |
10:48:08.357355+ DBCL USRC           8+ 1  "   +---------------------------+------------+------------+------------------+-----------
10:48:08.357370| DBCL USRC         307| 1  "   |                           |            |            |                  |
10:48:08.357685| DBCL USRC           2| 1  "   |                           |            |            |                  |
10:48:08.357690| DBCL USRC         121| 1  "   |                           |            |            |                  |
10:48:08.357818+ DBCL USRC           1+ 1  "   +---------------------------+------------+------------+------------------+-----------
10:48:08.357822| DBCL USRC           2| 1  "   |                           |            |            |                  |
10:48:08.357835| MPUT NT -------------| 1  "   |                           |            |            |                  |
10:48:08.357850| MPUT NT -------------| 1  "   |                           |            |            |                  |
10:48:08.357859+ MPUT NT -------------+ 1  "   +---------------------------+------------+------------+------------------+-----------
10:48:08.357868| PEND FI -------------| 1  "   |                           |            |            |                  |
10:48:08.358905| strt        >>>WRUDS | 1 8E83 |    1   192111      1     33|       41722 |      100624 | 0 *            0 |          0
10:48:08.358975| INIT    -------------| 1  "   |                           |            |            |                  |
10:48:08.358986+ MGET    -------------+ 1  "   +---------------------------+------------+------------+------------------+-----------
10:48:08.359007| DBCL USRC           7| 1  "   |                           |            |            |                  |
10:48:08.359018| DBCL USRC         707| 1  "   |                           |            |            |                  |
10:48:08.359214| strt        >>>WRUDS | 2 8E86 |    1   192112      1     30|       40893 |       98246 | 0 *            0 |          0
10:48:08.359274+ INIT    -------------+ 2  "   +---------------------------+------------+------------+------------------+-----------
10:48:08.359286| MGET    -------------| 2  "   |                           |            |            |                  |
10:48:08.359304| DBCL USRC           6| 2  "   |                           |            |            |                  |
10:48:08.359317| DBCL USRC         173| 2  "   |                           |            |            |                  |
10:48:08.359496+ DBCL USRC           1+ 2  "   +---------------------------+------------+------------+------------------+-----------
10:48:08.359500| DBCL USRC          41| 2  "   |                           |            |            |                  |
10:48:08.359544| DBCL USRC        1188| 2  "   |                           |            |            |                  |
10:48:08.359614| RSET    -------------| 1 8E8R |    1   192109      1     34|            |            |                  |
10:48:08.359653+ DBCL CATA         192+ 1  "   +    1   192109      1     34+ ----------- +------------+------------------+-----------
10:48:08.359734| DBCL USRC           1| 1 8E83 |                           |            |            |                  |
10:48:08.359738| DBCL USRC         129| 1  "   |                           |            |            |                  |
10:48:08.359850| DBCL EDVG          87| 1 8E8R |                           |            |            |                  |
10:48:08.359872+ DBCL USRC        2389+ 1 8E83 +---------------------------+------------+------------+------------------+-----------
10:48:08.359945| CONT 4C -------------| 1 8E8R |    1   192109      1     34|            |            |                  |
10:48:08.359992| DBCL USRC           5| 1  "   |                           |            |            |                  |
10:48:08.360002| DBCL USRC         251| 1  "   |                           |            |            |                  |
```

```
10:48:08.360262+ DBCL USRC         111+ 1  "   +----------------------------+-------------+-------------+------------------+-----------
10:48:08.360378| DBCL USRC          34| 1  "   |                            |             |             |                  |
10:48:08.360417| DBCL USRC           2| 1  "   |                            |             |             |                  |
10:48:08.360423| DBCL USRC        4689| 1  "   |                            |             |             |                  |
10:48:08.360740+ DBCL USRC           2+ 2 8E86 +----------------------------+-------------+-------------+------------------+-----------
10:48:08.360745| DBCL USRC           2| 2  "   |                            |             |             |                  |
10:48:08.360757| MPUT NT -------------| 2  "   |                            |             |             |                  |
10:48:08.360777| MPUT NT -------------| 2  "   |                            |             |             |                  |
10:48:08.360785+ MPUT NT -------------+ 2  "   +----------------------------+-------------+-------------+------------------+-----------
10:48:08.360795| PEND FI -------------| 2  "   |                            |             |             |                  |
10:48:08.361494| DBCL FITA         574| 1 8E85 |    1   192110      1     32|             |             |                  |
10:48:08.362080| CONT 31 -------------| 1  "   |                            |             |             |                  |
```

14 Lastsimulation mit Workload Capture and Replay

Mit der Funktion Workload Capture & Replay kann die Kommunikation von UTM-Anwendungen mit UPIC-Clients mitgeschnitten und anschließend mit einstellbaren Lastprofilen abgespielt werden. Damit lässt sich das Verhalten der UTM-Anwendung bei hoher Last unter realen Bedingungen testen.

Workload Capture & Replay besteht aus folgenden Komponenten:

- *UPIC Capture*: schneidet die Kommunikation mit dem UPIC-Client mit.

  Zum Mitschneiden von UPIC-Sessions (Capture) wird die Trace-Funktion BTRACE (BCAM-Trace) verwendet, die auf allen Server-Plattformen vorhanden ist.

  Die Traces müssen ggf. noch zusammengemischt werden.
- *UPIC Analyzer*: dient zur Analyse der mitgeschnittenen Kommunikation.

  Zur Analyse wird das Programm *UPICAnalyzer* verwendet, das mit UPIC auf 64-Bit-Linux-Systemen ausgeliefert wird.
- *UPIC Replay*: dient zum Abspielen der mitgeschnittenen UPIC-Session mit unterschiedlichen Lastparametern (Geschwindigkeit, Client-Anzahl).

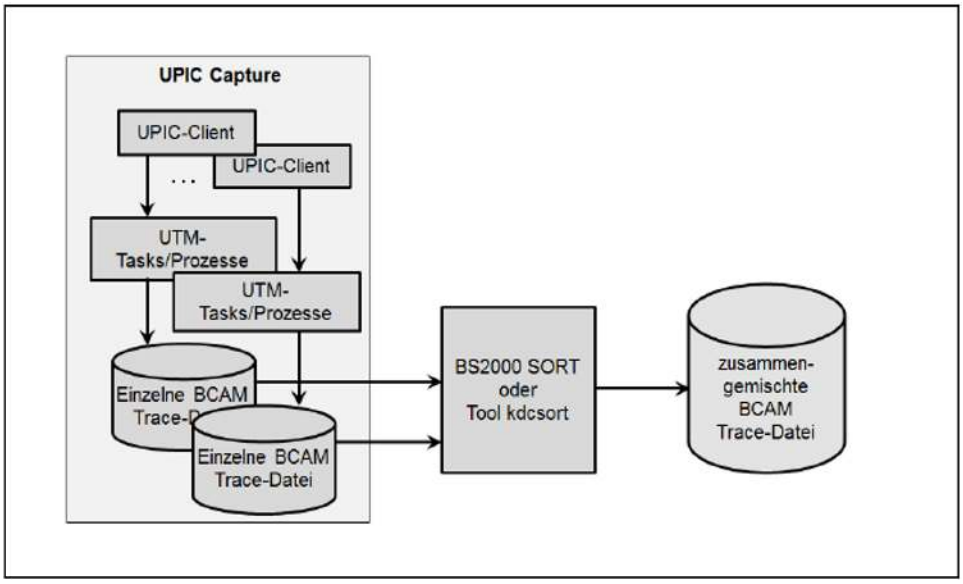
  Dazu wird das Programm *UPICReplay* verwendet, das mit UPIC auf 64-Bit-Linux-Systemen ausgeliefert wird.

> ¡ Das *UpicAnalyzer* Programm muss zu der UTM-Version kompatibel sein, die beim Capture-Vorgang verwendet wurde. Z.B. ist „openUTM-Client V7.0 für Trägersystem UPIC“ kompatibel zu openUTM V7.0.
>
> Die Version des *UpicReplay* Programms kann nur Eingabedateien verarbeiten, die mit der gleichen Version des *UpicAnalyzer* Programms erstellt wurden.

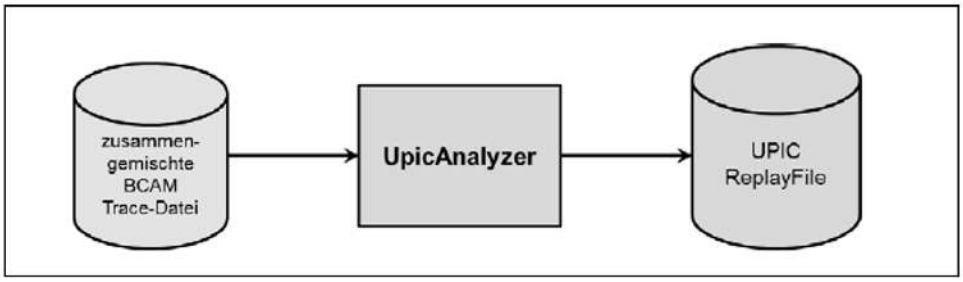
Die Funktion Workload Capture & Replay führen Sie in folgenden Schritten durch:

1. Schalten Sie den BCAM-Trace ein und starten Sie die UPIC-Kommunikation, siehe Abschnitt „UPIC-Conversation mitschneiden (UPIC Capture)“.
2. Beenden Sie den BCAM-Trace und mischen Sie die BCAM-Trace-Einträge in eine Trace-Datei zusammen (falls nötig), siehe Abschnitt „Trace-Einträge zusammenmischen“.

   Beide Schritte werden in folgendem Bild veranschaulicht.

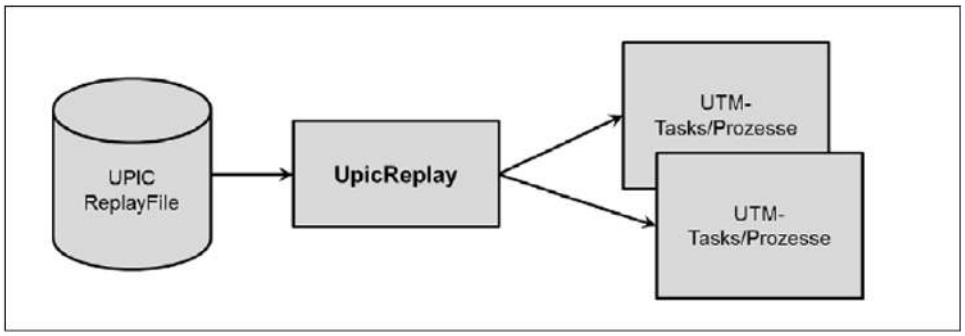


3. Übertragen Sie die Trace-Datei binär auf das 64-Bit Linux-System, auf dem Sie UPIC installiert haben. Bitte beachten Sie, dass Sie die Datei mit openFT übertragen müssen.
4. Erzeugen Sie auf dem 64-Bit Linux-System mit installierten UPIC-Client eine UPIC ReplayFile. Dazu rufen Sie das Programm *UpicAnalyzer* auf mit der Trace-Datei als Eingabedatei, siehe Bild. Details siehe Abschnitt „Daten mit dem Programm UpicAnalyzer aufbereiten“.



5. Starten Sie das Programm

   UpicReplay mit UPIC ReplayFile als Eingabedatei, siehe Bild. Details siehe Abschnitt „UPIC-Session mit dem Programm UpicReplay abspielen“.



14.1 UPIC-Conversation mitschneiden (UPIC Capture)

Bei diesem Schritt kann die UTM-Anwendung auf einer beliebigen UTM-Plattform ablaufen (BS2000-, Unix-, Linux- oder Windows-System).

Die UPIC-Clients können auf jeder beliebigen UPIC-Plattform ablaufen, auch UPIC-Clients auf Basis des Produkts openUTM-JConnect werden unterstützt.

Während dieser Phase muss die Kommunikation der UTM-Anwendung mit UPIC-Clients vollständig aufgezeichnet werden, wobei die Trace-Länge größer als die maximale Nachrichtenlänge sein muss. Hierfür wird die UTM-Funktion BCAM-Trace verwendet.

> ¡ Auf BS2000 unterstützt sowohl der UPIC-Client als auch die UTM-Anwendung Encryption,. wenn dies auf dem B2000-System installiert ist. In diesem Fall kann nicht ohne Verschlüsselung gearbeitet werden.
>
> Außerdem ignoriert *UpicReplay* den Aufruf des UPIC-Clients zum Setzen eines neuen Passworts.

Bitte beachten Sie, dass die ausgesuchten UTM-Vorgänge auch beliebig oft wiederholbar sein müssen.

Dazu gehen Sie wie folgt vor:

1. Starten Sie den BCAM-Trace per Startparameter BTRACE=ON,*length*, siehe "Startparameter der Anwendung". Es wird empfohlen, für *length* den Maximalwert anzugeben, damit Nachrichten nicht abgeschnitten werden. Sie können den BCAM-Trace auch per Administration einschalten (Kommando KDCDIAG oder über WinAdmin /WebAdmin). In diesem Fall wird aber für *length* der Standardwert (256 Bytes) angenommen, wenn Sie im Startparameter BTRA-CE keine andere Länge angegeben haben.
2. Führen Sie die für die Lastsimulation benötigten UPIC-Conversations zwischen dem UPIC-Client und der UTM-Anwendung aus. Dazu gehört auch der vollständige Verbindungsaufbau der UPIC-Clients. Die zugehörigen UTM-Vorgänge müssen mindestens einmal komplett durchlaufen werden.
3. Beenden Sie den BCAM-Trace per Kommando KDCDIAG oder über WinAdmin/WebAdmin.

Das Ergebnis dieses Schrittes sind binäre BTRACE-Dateien von allen UTM-Prozessen. Details zu BTRACE-Dateien finden Sie im openUTM-Handbuch „Meldungen, Test und Diagnose“.

14.2 Trace-Einträge zusammenmischen

Dieser Schritt ist nötig, falls die UTM-Anwendung beim Mitschneiden mit mehr als einem Prozess gelaufen ist, was bei produktiven UTM-Anwendungen mit mittlerer oder hoher Last in der Regel der Fall ist.

Die binären BTRACE-Dateien von allen UTM-Prozessen werden in diesem Schritt auf Basis der Zeitstempel in eine gemeinsame BTRACE-Datei einsortiert. Dieser Prozess-Schritt muss immer auf der gleichen Plattform wie Schritt 1 (UPIC Capture) ablaufen.

Auf BS2000-Systemen verwenden Sie dazu das BS2000-Dienstprogramm SORT.

Das Ergebnis dieses Schrittes ist eine sortierte binäre BTRACE-Datei, die alle Trace-Einträge in der zeitlich korrekten Reihenfolge enthält.

Sie können zum Sortieren die ausgelieferte Beispiel-Prozedur BTRACE verwenden, siehe auch Abschnitt „Beispielprozeduren“.

14.3 Daten mit dem Programm UpicAnalyzer aufbereiten

Das Programm *UpicAnalyzer* wird mit UPIC auf Linux (64 Bit) ausgeliefert. *UpicAnalyzer* liest die Trace-Records aus einer BTRACE-Trace, filtert die UPIC-Trace-Records aus, bereitet diese auf und schreibt sie in einem bestimmten Format (UPIC ReplayFile Layout) in eine Datei. Diese Datei kann dann als Eingabedatei für das Programm *UpicReplay* verwendet werden.

UpicAnalyzer wird wie folgt aus der Linux-Shell aufgerufen:

`UpicAnalyzer` *inputfile outputfile*

Bedeutung der Parameter:

inputfile

Name der BTRACE-Datei, die Sie auf das Linux-System übertragen haben.

outputfile

Name der Ausgabedatei (UPIC ReplayFile). Diese Datei können Sie verwenden, um die UPIC-Session mit Hilfe von *UpicReplay* ablaufen zu lassen.

Das Programm *UpicAnalyzer* erkennt den Plattform-Typ, auf dem die Trace-Datei erstellt wurde, und verarbeitet den Inhalt entsprechend der Plattform-spezifischen Besonderheiten.

Beispiel

Die übertragene Trace-Datei hat den Namen *btrc.sorted*. Sie soll aufbereitet und die Ausgabe in die Datei *Replayfile* geschrieben werden. Der Aufruf lautet:

```
UpicAnalyzer btrc.sorted Replayfile
```

Ausgaben:

```
Program "UpicAnalyzer": Version 7.0 build yyyy-mm-dd on Linux Intel, 64 Bit, Little-Endian started

inputfile "btrc.sorted"

outputfile "Replayfile"

109 UTM BCAM trace records with 17218 bytes read.

25 UPIC replay records with 2046 bytes written.

Program "UpicAnalyzer" finished.
```

14.4 UPIC-Session mit dem Programm UpicReplay abspielen

Das Programm *UpicReplay* ist ein UPIC-Client-Programm, das mit UPIC auf Linux (64 Bit) ausgeliefert wird. Vor dem Abspielen müssen Sie ggf. die UPIC-Konfiguration und/oder die Generierung der UTM-Anwendung anpassen.

Beim Abspielen sollte die gleiche UTM-Plattform verwendet werden wie beim Mitschneiden. Ausnahmen sind möglich, siehe „Unterschiedliche Plattformen für Capture and Replay“.

14.4.1 UPIC-Konfiguration und UTM-Generierung anpassen

Für den Ablauf auf dem Linux-System wird die Side Information-Datei *upicfile* benötigt, in der mindestens ein Eintrag mit dem Namen UPREPLAY zu finden ist. Der Eintrag muss das Präfix SD oder ND haben, Ausnahme siehe „Unterschiedliche Plattformen für Capture and Replay“.

Dieser Eintrag muss ein gültiger Eintrag mit dem TAC eines Services der UTM-Anwendung sein. (z.B. "DEMO"). Dieser Eintrag wird vom Programm *UpicReplay* zur Adressierung der UTM-Anwendung verwendet. Der TAC wird ggf. vom Programm *UpicReplay* passend mit Daten aus dem Replay File gesetzt.

Beispiel für einen upicfile-Eintrag

Replay mit dem TAC DEMO. Die UTM-Anwendung UTMTEST1 läuft auf dem Rechner HOST5678.

```
SDUPREPLAY UTMTEST1.HOST5678 DEMO LISTENER-PORT=102 T-TSEL-Format=T
```

UTMTEST1 muss in einer BCAMAPPL-Anweisung generiert sein.

Hinweise zur UTM-Generierung

Die UTM-Anwendung muss beim UPIC Replay Schritt, insbesondere bei erhöhter Last, möglicherweise mehr UPIC-Verbindungen vom Programm *UpicReplay* zulassen als während des Mitschneidens ursprünglich vorhanden waren. Daher wird empfohlen, für den UPIC-Zugang einen ausreichend dimensionierten UPIC Terminal-Pool mit Multi-Connect Funktionalität zu verwenden, z.B.:

```
TPOOL LTERM=REPL,PTYPE=UPIC-R,CONNECT=MULTI,NUMBER=1000
```

In diesem Fall können sich bis zu 1000 UPIC-Clients gleichzeitig über den Terminal-Pool anmelden.

Wenn der UPIC Replay Schritt mit erhöhter Last abläuft, dann müssen lastabhängige Generierungsparameter ggf. vergrößert werden. Insbesondere müssen Sie auf Folgendes achten:

- Ausreichende Größe des UTM-Cache (MAX CACHESIZE)
- Ausreichende Größe des Page Pool (MAX PGPOOL)
- Ausreichende Anzahl der UTM-Tasks (MAX TASKS)
- Ausreichende Anzahl der zugelassenen Concurrent User (MAX CONN-USERS)

Unterschiedliche Plattformen für Capture and Replay

Beim Abspielen werden die Daten 1:1 an die UTM-Anwendung übergeben. Wenn die Daten z.B. hardware-abhängige Binärdaten enthalten, dann führt dies beim Plattformwechsel zu Fehlern. Daher gilt Folgendes:

- Es ist nicht möglich, eine Session mit einer UTM-Anwendung auf BS2000-Systemen mitzuschneiden und später mit einer UTM-Anwendung auf Unix-, Linux-, oder Windows-Systemen abzuspielen. Der Grund: Die Daten liegen in der Trace-Datei in EBCDIC vor, eine Umcodierung nach ASCII wird in UPIC nicht unterstützt.
- Ein Wechsel zwischen einer 32-Bit- und einer 64-Bit-Plattform ist nicht möglich, auch nicht innerhalb einer Plattformfamilie.
- Es ist möglich, eine Session mit einer UTM-Anwendung auf Unix-, Linux-, oder Windows-Systemen mitzuschneiden und später mit einer UTM-Anwendung auf BS2000-Systemen abzuspielen. Voraussetzung ist, dass in der Session nur reine ASCII-Textdaten übertragen werden.

  In diesem Fall müssen Sie in der Datei *upicfile* HD als Präfix angeben, damit die Daten korrekt zwischen ASCII und EBCDIC umcodiert werden.

14.4.2 Aufruf von UpicReplay

UpicReplay spielt die aufgezeichneten UPIC-Conversations erneut ab, siehe Abschnitt „Arbeitsweise von UpicReplay“. Protokoll-Meldungen und Warnungen werden dabei nach *stdout* und Debug- oder Fehler-Meldungen nach *stderr* ausgegeben.

UpicReplay wird wie folgt aus einer Linux-Shell aufgerufen:

`UpicReplay` *InputFileName* [`-c`<*numberOfClients*>] [`-s`<*speedPercentage*>] [`-d`[d]]

Bedeutung der Parameter

InputFileName

Name des UPIC ReplayFile, das Sie mit dem UpicAnalyzer erzeugt haben.

Pflichtparameter.

-c<numberOfClients>

numberOfClients gibt die Anzahl der UPIC-Clients an, für welche die aufgezeichneten Conversations abgespielt werden sollen. Es können maximal so viele Clients wie aufgezeichnet wurden erneut abgespielt werden.

Standard: 1, (entspricht *-c1*) d.h. es wird nur ein Client simuliert.
Das effektive Limit hängt von den jeweiligen System-Limits ab

-s<speedPercentage>

speedPercentage gibt die Abspielgeschwindigkeit in Prozent im Vergleich zur Originalgeschwindigkeit an. Damit lassen sich lange und kurze Denkzeiten simulieren.

Standard: 100 (entspricht *-s100)* d.h. Originalgeschwindigkeit

-s200 bedeutet 200%, d.h. doppelte Geschwindigkeit, realisiert durch halbe Denkzeiten.

-d aktiviert Debug-Ausgaben auf *stderr*, d.h. Ausgabe von Debug-Meldungen bei Thread-Erzeugung sowie wenige Meldungen bei Send- und Receive-Aufrufen.

-dd

aktiviert erweiterte Debug-Ausgaben auf *stderr*, d.h. Ausgabe von detaillierten Debug-Meldungen. Diese Option ist nur für die interne Diagnose von *UpicReplay* gedacht.

-dd ist nur sinnvoll bei Simulation einer kleinen Anzahl von Clients.

Standard: keine Debug-Ausgaben.

Beispiel

Die in der Datei *Replay.1239* aufgezeichneten UPIC Conversations sollen mit normaler Geschwindigkeit für maximakl 100 Clients abgespielt werden. Der Aufruf lautet:

```
UpicReplay Replay.1239 -c100
```

14.4.3 Arbeitsweise von UpicReplay

UpicReplay spielt die Kommunikation möglichst 1:1 wie beim aufgezeichneten Ablauf nach:

- Für jedes UPIC PTERM/LTERM, für das in dem UPIC ReplayFile ein Trace-Record gefunden wird, wird ein UPIC-Thread erzeugt, der die jeweilige UPIC-Conversation dieses UPIC-Clients nachspielt.
- Dieser UPIC-Thread schickt in Schleife alle Eingabe-Nachrichten in gleicher Weise an den UTM-Vorgang wie beim Mitschneiden, d.h. mit dem gleichen Daten-Inhalt und Kontrollfluss. Analoges gilt für das Holen der Ausgabe-Nachrichten von der UTM-Anwendung, wobei die Ausgabe-Nachrichten inhaltlich nicht geprüft werden.

Probleme beim Replay

In den folgenden Fällen kommt es zu Abweichungen zwischen dem Mitschnitt und der Wiederholung beim Replay:

- Unvollständiger Mitschnitt

  Eine UPIC-Conversation (d.h. ein UTM-Vorgang) wurde begonnen, bevor das Mitschneiden per BCAM-Trace eingeschaltet wurde.

  Es wird eine entsprechende Meldung ausgegeben und alle Eingabe-Nachrichten aus dieser begonnenen Conversation dieses Clients werden verworfen.

  Der UPIC-Thread sucht dann im Mitschnitt nach dem Beginn einer neuen Conversation für diesen UPIC-Client:

  - Wird eine neue Conversation in den aufgezeichneten Records für dieses PTERM/LTERM gefunden, dann wartet dieser Client zunächst noch entsprechend den aufgezeichneten Zeitstempeln und beginnt dann erst ab dieser Stelle die Last-Simulation.
  - Wird keine neue Conversation gefunden, wird dieser UPIC Replay Thread ohne Kommunikation mit der UTM-Anwendung beendet.
- Verkürzter Vorgang

  Ein UTM-Vorgang wird beim Nachspielen nach weniger Kommunikations-Schritten als beim Mitschneiden von der UTM-Anwendung beendet (Normal oder Abnormal). Dieser Fall kann auftreten:

  - wenn das Anwendungsprogramm die mitgeschnittenen Eingabe-Daten nicht korrekt verarbeiten kann, weil z. B. Zeitangaben in der Eingabe-Nachricht stehen, die vom Programm als "verspätet" abgelehnt werden. Der UTM-Vorgang wird deshalb vorzeitig beendet.
  - wenn beim Nachspielen eine nicht zugelassene UTM-Zugangsberechtigung verwendet wird, z.B. fehlende UTM-Administrationsberechtigung.

  In diesem Fall wird eine entsprechende Meldung ausgegeben und der UPIC Replay Thread verwirft weitere Nachrichten, bis er wieder einen neuen Conversation-Beginn von diesem Client im Mitschnitt findet. An diesem Conversation-Beginn setzt der UPIC-Thread nach einer entsprechenden Verzögerungszeit neu auf oder er beendet sich, wenn keine weitere Conversation für diesen UPIC-Client aufgezeichnet wurde.
- Überlanger Vorgang

  Ein UTM-Vorgang hat beim Nachspielen mehr Kommunikations-Schritte als beim Mitschneiden.

  Wegen nicht mitgeschriebener Eingabe-Daten beendet der UPIC-Thread diesen Vorgang abnormal durch Verbindungs-Abbau. Zusätzlich wird eine spezifische Warnmeldung erzeugt.

  Anschließend wird der Beginn der nächsten Conversation von diesem Client im Mitschnitt gesucht. An diesem Conversation-Beginn setzt der UPIC-Thread nach einer entsprechenden Verzögerungszeit neu auf oder er beendet sich, wenn keine weitere Conversation für diesen UPIC-Client aufgezeichnet wurde.

- Eingabe-Nachricht unvollständig

  Eine Eingabe-Nachricht konnte wegen Längenbeschränkung des Trace-Records trotz Kompressionsversuchs beim Mitschneiden nicht vollständig mitgeschrieben werden.

  Der Record wird mit Warnmeldung verworfen und es wird der Beginn der nächsten Conversation von diesem Client im Mitschnitt gesucht.

  An diesem Conversation-Beginn setzt der UPIC-Thread nach einer entsprechenden Verzögerungszeit neu auf oder er beendet sich, wenn keine weitere Conversation für diesen UPIC-Client aufgezeichnet wurde.

- Sonstiger Fehler

  Es wird ein sonstiger, unerwarteter Return Code an der UPIC-Programmschnittstelle gemeldet, der nicht in den obigen Fällen enthalten ist.

  Diese Situation kann z.B. dann eintreten, wenn die UTM-Anwendung entweder nicht erreichbar ist oder einen Verbindungsaufbau ablehnt.

  In diesen Fällen wird vom UPIC-Thread eine Fehlermeldung ausgegeben.

  Der betroffene UPIC-Thread wird beendet, ohne nach neuen Conversations für diesen Client im Mitschnitt zu suchen. Alle von dem Problem nicht direkt betroffenen anderen UPIC-Conversations laufen unbeeinflusst weiter.

15 Anhang

- XS-Unterstützung von UTM-Anwendungen

15.1 openUTM installieren

Die Installation von openUTM erfolgt üblicherweise mit dem Verfahren IMON. openUTM wird als Subsystem des BS2000-Betriebssystems geladen. Dabei können verschiedene Versionen von openUTM als voneinander unabhängige Subsysteme UTM geladen werden.

Wenn Sie die Performance Ihrer Anwendungen mit SM2 oder KDCMON überwachen wollen, müssen Sie neben dem Subsystem UTM das Subsystem UTM-SM2 oder das Subsystem KDCMON laden.

Im Folgenden sind spezielle Informationen über openUTM, UTM-SM2 und KDCMON zusammengefasst, die für die Installation von Bedeutung sind.

Versionsabhängigkeiten zu anderen Softwareprodukten wie Datenbanksysteme, FHS etc., sowie Angaben zu Compilerversionen und kompatiblen Laufzeitsystemen finden Sie in der Freigabemitteilung. Sie wird als druckaufbereitete Datei auf dem Produktband mit ausgeliefert. Angaben zu Compilerversionen und Laufzeitsystemen finden Sie auch im Abschnitt „Compiler-Versionen, Laufzeitsysteme, KDCDEF-Optionen“.

15.1.1 UTM-Systemcode

Bestandteile des UTM-Systemcodes

Der UTM-Systemcode besteht aus folgenden Teilen:

- dem Basissystem
- dem Abbildungsmodul für die jeweilige BS2000-Version
- den Modulen für UTM-D-Funktionen
- den Schnittstellen-Modulen für die Verschlüsselungsfunktionen

Diese Bestandteile werden beim Laden als getrennte Einheiten behandelt und in der angegebenen Reihenfolge geladen.

Abbildungsmodule - Ablauf von openUTM unter verschiedenen BS2000-Versionen

Die Module des UTM-Basissystems sind unabhängig von der Betriebssystemversion. Wenn openUTM Funktionen des Betriebssystems braucht, dann ruft openUTM diese über „neutrale Schnittstellen“ auf. Die Abbildung der neutralen Schnittstellen auf System-spezifische Schnittstellen erfolgt in einem Abbildungsmodul. Diese Technik erlaubt es, eine openUTM-Version in verschiedenen Betriebssystemversionen einzusetzen.

Für jede BS2000-Version, in der openUTM ablauffähig ist, gibt es folgendes Abbildungsmodul (= Großmodul):

Abbildungsmodul	BS2000-Version	Plattform
KCYV190	BS2000 OSD/BC V10.0	/390 und x86
KCYV200	BS2000 OSD/BC V11.0	/390 und x86

Das zur BS2000-Version gehörende Abbildungsmodul wird aus der Bibliothek dynamisch nachgeladen, in der das UTM-Basissystem liegt. Die Verknüpfung zum Basissystem wird hergestellt.

Namensräume für Module des UTM-Systemcodes

Für die Objektmodule der Bestandteile des UTM-Systemcodes gelten eigene Namensräume. Die Namen für ENTRYs und EXTRNs beginnen mit:

KCS	im Basissystem
KCY	in den Abbildungsmodulen
KCD	in den UTM-D-Modulen
KCO	in XAP-TP-Modulen
KCE	in den Encryption Modulen (Verschlüsselung)

Auslieferung und Liefereinheiten

Alle Bestandteile für die Nutzung der UTM-D Funktionen werden als Teil der Liefereinheit openUTM V7.0 ausgeliefert. Für die Nutzung der UTM-D-Funktionen ist jedoch eine Lizenz erforderlich.

UTM-Systemcode

Der gesamte Systemcode von openUTM V7.0 wird in folgender Bibliothek ausgeliefert:

- SYSLNK.UTM.070.TPR (auf Servern mit /390-Architektur) bzw.
- SKMLNK.UTM.070.TPR (auf Servern mit x86-Architektur)

Diese Bibliothek enthält den Systemcode des Basissystems openUTM V7.0, die Abbildungsmodule für die BS2000-Versionen, den Systemcode der UTM-D-Module und die Schnittstelle zu den Verschlüsselungsfunktionen.

Die Objektkorrekturen für alle Module des UTM-Systemcodes befinden sich in der REP-Datei SYSREP.UTM.070. Die REP-Datei wird bei der Installation mit IMON aus der Lieferdatei SYSRMS.UTM.070 erzeugt. Die Korrekturen werden beim Laden des Subsystems UTM eingefügt.

Die NOREF-Datei SYSNRF.UTM.070 wird mitgeliefert. Die Datei enthält Namen von Modulen, die zur REP-Verarbeitung benötigt werden.

15.1.2 Produktdateien installieren

Die Produkt-spezifischen Parameter zur Installation von openUTM stehen in der ausgelieferten SYSII-Datei, hier ist für alle Produktdateien von openUTM *DEFAULT-USERID als Installationskennung eingestellt. Ein Teil der Parameter kann mit IMON-Mitteln an Kundenbedürfnisse angepasst werden.

Weitere Infos stehen im IMON-Manual.

Eine Auflistung der zu openUTM gehörenden Produktdateien finden Sie in der Freigabemitteilung.

15.1.3 UTM-Systemcode laden

Einträge im Subsystemkatalog für openUTM erzeugen

Der UTM-Systemcode läuft als ein Subsystem UTM des BS2000-Betriebssystems ab. Der gesamte UTM-Systemcode inklusive Abbildungsmodul und UTM-D-Modulen wird vom BS2000-Subsystemmanagement (DSSM) geladen.

Mit openUTM werden SSD-Objekte (Subsystem-Deklaration-Objekte) ausgeliefert. Mit diesen SSD-Objekten erzeugt IMON bei der Installation von openUTM im Subsystemkatalog einen Eintrag für das Subsystem UTM.

.

Folgende SSD-Objekte werden mit openUTM V7.0 ausgeliefert:

Datei	Einsatzfall
SYSSSC.UTM.070.190	openUTM auf BS2000 OSD/BC V10.0 (auf Servern mit /390- und x86-Architektur)
SYSSSC.UTM.070.200	openUTM auf BS2000 OSD/BC V11.0 (auf Servern mit /390- und x86-Architektur)

Die UTM-D-Module sind im Lieferumfang enthalten.

In dem Subsystemkatalog-Eintrag, den IMON mit diesen SSD-Objekten erzeugt, sind folgende Voreinstellungen wirksam:

- Ladezeitpunkt:
  Das Subsystem UTM muss explizit mit Kommando START-SUBSYSTEM geladen werden:
  ```
  /START-SUBSYSTEM SUBSYSTEM-NAME=UTM, VERSION='07.0'
  ```
- Installation-Userid: *DEFAULT-USERID
- Lade-Bibliothek (diese erfragt DSSM bei IMON, dort sind die folgenden Bibliotheken festgelegt:):
  SYSLNK.UTM.070.TPR auf Servern mit /390-Architektur
  SKMLNK.UTM.070.TPR auf Servern mit x86-Architektur
- REP-Datei: SYSREP.UTM.070

Diese Voreinstellungen für das UTM-Subsystem können vom Systemverwalter geändert werden. Details dazu finden Sie im Benutzerhandbuch "IMON".

15.1.4 UTM-Systemcode entladen

Der gesamte UTM-Systemcode, inklusive Abbildungsmodul und openUTM-D-Funktionen, kann nur entladen werden, wenn keine UTM-Anwendung mehr läuft. Zum Entladen des Systemcodes von openUTM muss der Systemverwalter das folgende Kommando eingeben:

```
/STOP-SUBSYSTEM SUBSYSTEM-NAME=UTM, VERSION='07.0'
```

Nach dem Entladen kann der UTM-Systemcode erneut geladen werden, d.h. der UTM-Systemcode kann bei laufendem BS2000-System komplett ausgetauscht werden. Diese Funktion können Sie zum Umsteigen auf eine UTM-Korrekturversion oder auf eine neue openUTM-Version nutzen.

Das erneute Laden kann mit Hilfe des Kommandos START-SUBSYSTEM veranlasst werden. Haben Sie im Subsystemkatalog-Eintrag für openUTM angegeben, dass der UTM-Systemcode beim ersten Start einer UTM-Anwendung geladen wird, so löst der nächste Start einer UTM-Anwendung das erneute Laden des UTM-Systemcodes aus.

15.1.5 Meldungsdateien

Für die Ausgabe von UTM-Meldungen über die BS2000-Meldungsverarbeitung wird die Meldungsdatei SYSMES.UTM.070 mit openUTM ausgeliefert. Bei der Installation mit IMON wird diese Datei automatisch als zusätzliche system-globale Meldungsdatei aktiviert.

15.1.6 REP-Dateien und RMS-Dateien

Objektkorrekturen werden in Form einer RMS-Datei ausgeliefert. IMON erzeugt daraus bei der Installation direkt mit RMS eine REP-Datei. Für openUTM V7.0, UTM-SM2 und KDCMON werden die folgenden RMS-Dateien ausgeliefert:

Subsystem	**RMS-Datei**	**REP-Datei**
UTM	SYSRMS.UTM.070	SYSREP.UTM.070
UTM-SM2	SYSRMS.UTM-SM2.*nnn*	SYSREP.UTM-SM2.*nnn*
KDCMON	SYSRMS.KDCMON.*nnn*	SYSREP.KDCMON.*nnn*

Dabei hängt *nnn* wie folgt von der BS2000-Version ab:
nnn=190 für BS2000 OSD/BC V10.0
nnn=200 für BS2000 OSD/BC V11.0

15.1.7 Paralleler Betrieb mehrerer openUTM-Versionen

Es ist möglich, im selben BS2000-Betriebssystem mehrere openUTM-Versionen nebeneinander zu laden und gleichzeitig einzusetzen.

Diese Funktion ist besonders beim Übergang auf eine neue openUTM-Version von Vorteil. Sie können dann neben dem Produktivbetrieb einzelne UTM-Anwendungen versuchsweise mit der UTM-Folgeversion ablaufen lassen. Die Umstellung mehrerer UTM-Anwendungen von einer openUTM-Version auf eine andere innerhalb eines Rechners kann damit schrittweise getestet und durchgeführt werden.

Beim parallelen Betrieb mehrerer openUTM-Versionen müssen Sie Folgendes beachten:

- Im BS2000-Subsystemkatalog sind alle openUTM-Versionen einzutragen. Die Einträge müssen den Parallelbetrieb mehrerer openUTM-Versionen erlauben. Die mit openUTM ausgelieferten SSD-Objekte enthalten diese Erlaubnis.
- Beim Laden des UTM-Systemcodes mit dem Kommando `START-SUBSYSTEM` müssen Sie den Parameter `VERSION-PARALLELISM=*COEXISTENCE-MODE` angeben.
- Sämtliche Teile einer Anwendung müssen mit den Dateien/Bibliotheken der gleichen openUTM-Version erstellt werden.
- UTM-Anwendungen in verschiedenen openUTM-Versionen dürfen nicht den gleichen Namen haben. openUTM verhindert den Start der zweiten Anwendung gleichen Namens.

15.1.8 Subsystem UTM-SM2

Der Messmonitor SM2 kann Performancewerte zu laufenden UTM-Anwendungen in der SM2-Messdatei ablegen und am Bildschirm ausgeben. Voraussetzung ist, dass der Baustein UTM-SM2 installiert ist. UTM-SM2 ist in BS2000-GA (Grundausbau) enthalten. UTM-SM2 wird als eigenständiges Subsystem des BS2000 geladen.

Bestandteile

Zu UTM-SM2 gehören die im Folgenden aufgelisteten Bibliotheken und Dateien:

Name	Inhalt
SYSLNK.UTM-SM2.*nnn* SKMLNK.UTM-SM2.*nnn*	Ladebibliothek für Server mit /390-Architektur Ladebibliothek für Server mit x86-Architektur
SYSSSC.UTM-SM2.*nnn*	SSD-Objekt für BS2000-Betriebssystem
SYSREP.UTM-SM2.*nnn*	REP-Datei

Dabei bezeichnet *nnn* die UTM-SM2-Version:

nnn=190 für UTM-SM2 V19.0 auf BS2000 OSD/BC V10.0
nnn=200 für UTM-SM2 V20.0 auf BS2000 OSD/BC V11.0

Die REP-Datei wird bei der Installation mit IMON aus der Lieferdatei SYSRMS.UTM-SM2.*nnn* erzeugt.

Einträge in den Subsystemkatalog

Bei der Standard-Installation erzeugt IMON für UTM-SM2 automatisch den Eintrag im Subsystemkatalog.

In dem SSD-Objekt SYSSSC.UTM-SM2.*nnn* bzw. SPMSSC.UTM-SM2.*nnn* sind folgende Voreinstellungen wirksam:

- Ladezeitpunkt:
  Das Subsystem UTM-SM2 muss explizit mit Kommando START-SUBSYSTEM geladen werden.
- Installation-Userid:
  *DEFAULT-USERID
- Lade-Bibliothek (diese erfragt DSSM bei IMON, dort sind die folgenden Bibliotheken festgelegt):
  SYSLNK.UTM-SM2.*nnn* auf Servern mit /390-Architektur
  SKMLNK.UTM-SM2.*nnn* auf Servern mit x86-Architektur
- REP-Datei: SYSREP.UTM-SM2.*nnn*

Voreinstellungen für das UTM-SM2-Subsystem können vom Systemverwalter geändert werden. Damit Änderungen dauerhaft sind, sollten sie mit den Mitteln von IMON durchgeführt werden. Details dazu finden Sie im Benutzerhandbuch "IMON".

> ¡ Die Voreinstellung für den Ladezeitpunkt darf der Systemverwalter **nicht** ändern.

Laden des Subsystems UTM-SM2

UTM-SM2 kann sowohl per Systemverwalter-Kommando als auch durch openUTM geladen werden.

- Laden über Kommando:

  ```
  /START-SUBSYSTEM SUBSYSTEM-NAME=UTM-SM2
  ```

- Laden durch openUTM, wobei es zwei Möglichkeiten gibt:
  - UTM-SM2 wird automatisch beim Start der Anwendung geladen, falls diese mit MAX SM2=ON generiert ist.
  - UTM-SM2 wird beim Einschalten der Datenlieferung durch die Administration implizit geladen, wenn die Anwendung mit MAX SM2=OFF oder MAX SM2=ON generiert ist.

Entladen von UTM-SM2

Entladen wird UTM-SM2 mit dem Kommando

```
/STOP-SUBSYSTEM SUBSYSTEM-NAME=UTM-SM2
```

15.1.9 Subsystem KDCMON

Der UTM-Messmonitor KDCMON ist als eigenständiges Subsystem des BS2000 realisiert. Sie können mit KDCMON UTM-Anwendungen vermessen, die unter verschiedenen openUTM-Versionen ablaufen. KDCMON ist in BS2000-GA (Grundausbau) enthalten.

Lieferbestandteile

Zum Einsatz von KDCMON sind die im Folgenden aufgelisteten Bibliotheken, Dateien und Programme erforderlich:

Name	Inhalt
SYSLNK.KDCMON.*nnn* SKMLNK.KDCMON.*nnn*	Ladebibliothek für Server mit /390-Architektur Ladebibliothek für Server mit x86-Architektur
SYSSSC.KDCMON.*nnn*	SSD-Objekt für BS2000-Betriebssystem
SYSREP.KDCMON.*nnn*	REP-Datei
SYSMES.KDCMON.*nnn*	Meldungsdatei für BS2000-Betriebssystem
SYSPRG.KDCMON.*nnn*	Programm zum Starten von KDCMON
SYSPRG.KDCMON.*nnn*.KDCPMSM	Programm zum Umsetzen und Sortieren der Messdaten

Dabei bezeichnet *nnn* die KDCMON-Version:
nnn=190 für KDCMON V19.0 auf BS2000 OSD/BC V10.0
nnn=200 für KDCMON V20.0 auf BS2000 OSD/BC V11.0

Die REP-Datei wird bei der Installation mit IMON aus der Lieferdatei SYSRMS.KDCMON.*nnn* erzeugt.

Installieren der Produktdateien von KDCMON

Die Lieferbestandteile von KDCMON werden beim Installieren unter *DEFAULT-USERID abgelegt.

Einträge in den Subsystemkatalog

Bei der Standard-Installation erzeugt IMON für KDCMON automatisch den Eintrag im Subsystemkatalog.

In dem SSD-Objekt SYSSSC.KDCMON.*nnn* sind folgende Voreinstellungen wirksam:

- Ladezeitpunkt:

  KDCMON wird implizit beim ersten Aufruf geladen.

- Installation-Userid:
  *DEFAULT-USERID
- Ladebibliothek (diese erfragt DSSM bei IMON, dort sind die folgenden Bibliotheken festgelegt):
  SYSLNK.KDCMON.*nnn* auf Servern mit /390-Architektur
  SKMLNK.KDCMON.*nnn* auf Servern mit x86-Architektur
- REP-Datei: SYSREP.KDCMON2.*nnn*

Diese Voreinstellungen für das KDCMON-Subsystem können vom Systemverwalter geändert werden. Damit Änderungen dauerhaft sind, sollten sie mit den Mitteln von IMON durchgeführt werden. Details dazu finden Sie im Benutzerhandbuch "IMON".

> ¡ Die Voreinstellung für den Ladezeitpunkt darf der Systemverwalter **nicht** ändern.

Laden des Subsystems KDCMON

Das Subsystem KDCMON wird implizit beim Start des Programms SYSPRG.KDCMON.nnn gestartet, das Sie zum Vermessen von Anwendungen benötigen. Explizit können Sie das Subsystem KDCMON starten mit dem Kommando::

```
/START-SUBSYSTEM SUBSYSTEM-NAME=KDCMON
```

Entladen des Subsystems KDCMON

Das Subsystem KDCMON wird entladen mit dem DSSM-Kommando

```
/STOP-SUBSYSTEM SUBSYSTEM-NAME=KDCMON
```

Soll KDCMON entladen werden, wenn die Datenaufzeichnung in einer der UTM-Anwendungen noch nicht abgeschlossen ist, muss Folgendes angegeben werden:

```
/STOP-SUBSYSTEM SUBSYSTEM-NAME=KDCMON,FORCED=YES
```

Die Datenaufzeichnung wird hiermit beendet.

15.2 UTM-Tools aufrufen

Die UTM-Tools können Sie per START-EXECUTABLE-PROGRAM oder über eigene SDF-Kommandos starten.

15.2.1 UTM-Tools per START-EXECUTABLE-PROGRAM starten

Die UTM-Tools werden als LLMs erzeugt und in der Programmbibliothek SYSLNK.UTM.070.UTIL ausgeliefert. Jedes Tool rufen Sie wie folgt auf:

```
/START-EXECUTABLE-PROGRAM FROM-FILE= -
/ *LIBRARY-ELEMENT(LIBRARY=$user-id.SYSLNK.UTM.070.UTIL,ELEMENT-OR-SYMBOL=toolname)
```

toolname ist der Name des Tools, z.B. KDCDEF, und $*user-id* die Installations-Benutzerkennung (IMON-Installationspfad).

15.2.2 UTM-Tools über eigene SDF-Kommandos starten

Die Kommandos für die UTM-Tools sind im SDF-Anwendungsbereich UTM abgelegt.

Es stehen folgende Kommandos zur Verfügung:

```
START-CALLUTM
START-KDCBTRC
START-KDCCSYSL
START-KDCDEF
START-KDCDUMP
START-KDCEVAL
START-KDCMMOD
START-KDCMON
START-KDCMTXT
START-KDCPMSM
START-KDCPSYSL
START-KDCUPD
START-XATMIGEN
```

Gemeinsame Parameter für alle Kommandos:

MONJV =
Angabe einer Monitor-Jobvariablen zur Überwachung des Programmlaufs. Standard: *NONE

MONJV = *NONE
Es wird keine Monitor-Jobvariable verwendet.

CPU-LIMIT =
Maximale CPU-Zeit in Sekunden, die das Programm beim Ablauf verbrauchen darf. Standard: *JOB-REST

CPU-LIMIT = *JOB-REST
Die verbleibende CPU-Zeit soll für die Aufgabe verwendet werden.

CPU-LIMIT = <integer 1..32767 seconds>

Maximale CPU-Zeit in Sekunden, die das Programm beim Ablauf verbrauchen darf

VERSION =
Produktversion des Tools, das gestartet werden soll.

VERSION = *STD
Keine explizite Angabe der Produktversion.
Die Produktversion wird folgendermaßen ausgewählt:

1. Die mit dem Kommando /SELECT-PRODUCT-VERSION vorgegebene Version.
2. Die höchste mit IMON installierte openUTM-Version.

Geben Sie die Version folgendermaßen an:

```
<Produkt-Version> : Format mm.n<a<so>>
```

mm	Hauptversion 1..99
n	Nachtragsversion 0..9
a	Manual-Status A..Z (optional)
so	Korrektur-Status 00..99 (optional)

Geben Sie im geführten Dialog wiederholt ? ein, um diese Ausgabe zu erhalten, siehe auch „Beispiel 2“.

Die installierten openUTM-Versionen zeigen Sie folgendermaßen an:

```
/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=UTM(VERSION=*ALL), -
                        LOGICAL-IDENTIFIER=*NONE
```

Zusätzlicher Parameter für das Kommando START-CALLUTM:

TRANSPORT-SYSTEM =
Angabe des Transportsystems, das CALLUTM benutzt.
Standard: *SOCKET

TRANSPORT-SYSTEM = *SOCKET
Das Transport-System SOCKET wird benutzt.

TRANSPORT-SYSTEM = *CMX
Das Transport-System CMX wird benutzt.

```
Die vollständige Syntax aller UTM-Kommandos zeigen Sie mit den folgenden Kommandos an:
/SDF-SYNTAX-FILE = INSTALLATION-PATH(INSTALLATION-UNIT='UTM',LOGICAL-ID='SYSSDF',DEFAULT-
PATH-NAME='*UNKNOWN')
/START-SDF-A
//OPEN-SYNTAX-FILE &SDF-SYNTAX-FILE, MODE = *READ,TYPE=*SYSTEM
//SHOW OBJECT = *DOM(UTM),SIZE=*MAX,IMPL-INFO=*NO(LANGUAGE=D)
//END
```

Beispiel 1

Aufruf des Tools CALLUTM über eigenes SDF-Kommando:

```
/START-CALLUTM?
KOMMANDO   : START-CALLUTM
--------------------------------------------------------------------------------
TRANSPORT-SYSTEM     = *SOCKET
VERSION              = *STD
MONJV                = *NONE
CPU-LIMIT            = *JOB-REST
--------------------------------------------------------------------------------
NEXT = *EXECUTE
KEYS : F1=?   F3=*EXIT   F5=*REFRESH   F6=*EXIT-ALL   F9=REST-SDF-IN
       F11=*EXECUTE   F12=*CANCEL
```

Beispiel 2

Aufruf des Tools KDCDEF über eigenes SDF-Kommando:

Wenn Sie jetzt im Feld VERSION zweimal ein „?“ eingeben, dann erhalten Sie folgende Ausgabe:

```
/START-KDCDEF?
KOMMANDO    : START-KDCDEF
OPERANDEN   : VERSION=*STD
------------------------------------------------------------------------------
VERSION              = *STD
                       *STD oder Produkt-Version
                       Produkt-Version des Programms KDCDEF mm.n<a<so>>
                       Manual-Status a and Korrektur-Status so sind optional
                       - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                       <Produkt-Version> : Format mm.n<a<so>>
                        mm : Hauptversion 1..99
                        n : Nachtragsversion 0..9
                        a : Manual-Status A..Z
                        so : Korrektur-Status 00..99
MONJV                = *NONE
CPU-LIMIT            = *JOB-REST
------------------------------------------------------------------------------
NEXT = *EXECUTE
*EXECUTE"F3" / Next-cmd / *CONTINUE / *EXIT"K1" / *EXIT-ALL"F1" /     *TEST"F2"
```

15.3 Speicherklassen einer UTM-Anwendung

Den Speicherbedarf der UTM-Produktdateien und einer UTM-Anwendung entnehmen Sie der Freigabemitteilung.

In der folgenden Tabelle ist zusammengefasst, welche Bestandteile beim Betrieb von UTM-Anwendungen die Speicherklasse 4, 5 oder 6 benötigen:

Bestandteil	Klasse	Erläuterung
Systemcode openUTM	4	1.
statische KAA-Tabellen	5	2.
dynamische KAA-Tabellen	5	3.
Cache-Speicher	(5)	4.
KTA-Tabellen	5	5.
ROOT-Systemmodule	6	
Zusatzbedarf bei Formatierung	6	6.
Zusatzbedarf bei DB-System	6	7.
Administrationsprogramm KDCADM	6	
Teilprogramme des Anwenders	6	8.

Erläuterung

1. Systemcode openUTM:
   Der UTM-Systemcode wird durch das BS2000-Subsystemmanagement DSSM in den Klasse-4-Speicher geladen. Zum UTM-Systemcode gehören die Komponenten UTM-Basis inklusive Betriebssystem-Abbildungsmodul und der UTM-D-Code inklusive XAP-TP. Einzelheiten dazu finden Sie im Abschnitt „openUTM installieren“.
2. Mit der Meldung `K450 KDCFILE generated; KAA-size:nnnK;...` protokolliert KDCDEF bei der Generierung, dass die statischen KAA-Tabellen *nnn* KB belegen. Beim Start der Anwendung liest openUTM diese Tabellen in einen Memory Pool im Klasse-5-Speicher ein. Der Memory Pool belegt *nnn* KB, aufgerundet auf ein Vielfaches von 1 MB.

   Die Hinzunahme eines Benutzers und eines Terminals verursacht einen Mehrbedarf von (mindestens) 1 KB. Für „Extras“ wie Kellerung, TLS, ULS etc. steigt dieser Wert. Für eine Abschätzung des Platzbedarfs durch Erweiterung der Konfiguration ist es besser, die Anzahl der Objekte mit RESERVE-Anweisungen in den KDCDEF-Input einzutragen und KDCDEF den Platzbedarf selbst ausrechnen zu lassen.
3. Dynamische KAA-Tabellen:
   Der Platzbedarf für ein Terminal ist mindestens 1 KB. openUTM nutzt zunächst den Verschnitt am Ende des Memory Pools, der für die statischen KAA-Tabellen angelegt wurde. Wenn dieser Platz nicht ausreicht, legt openUTM einen oder mehrere weitere Memory Pools für die dynamischen Tabellen im Klasse-5-Speicher an.

4. Cache-Speicher:
   Als Pufferbereich für I/O's zum Pagepool auf KDCFILE benützen alle Prozesse einer UTM-Anwendung den Cache-Speicher. openUTM legt ihn entweder als Memory Pool im Klasse-5-Speicher des Programmraum oder in einem oder mehreren Datenräumen an. Durch das Anlegen des Cache-Speichers in einem Datenraum kann der Programmraum entlastet werden. Ablageort und Größe des Cache-Speichers werden durch die KDCDEF-Parameter MAX CACHESIZE=(*number*,..., PS | DS) und BLKSIZE=*blocksize* bestimmt.

   Beispiel
   Bei *blocksize*=2K ist er *number* * 2KB groß, aufgerundet auf ein Vielfaches von 1 MB.

5. KTA-Tabellen (**K**DC **T**ask **A**rea):
   Der Bereich enthält Prozess-spezifische Verwaltungsdaten, er wird im Klasse-5-Speicher angelegt. Die Länge ist mindestens 8KB. Der Platzbedarf steigt in Abhängigkeit von den KDCDEF-Generierungsparametern.

6. Zusatzbedarf bei Formatierung:
   Der Code des Formatierungssystems wird nachgeladen. Zum Speicherbedarf für das Formatierungssystem siehe Angaben von FHS. Dazu kommt der Speicherbedarf für die Formate.

7. Zusatzbedarf bei Datenbanksystem:
   Das Verbindungsmodul des Datenbanksystems kann statisch oder dynamisch dazugebunden werden. Den Speicherbedarf entnehmen Sie den Informationen des jeweiligen Datenbanksystems.

8. Teilprogramme des Anwenders:
   Den Speicherbedarf entnehmen Sie der Binderliste.

15.4 Compiler-Versionen, Laufzeitsysteme, KDCDEF-Optionen

In diesem Abschnitt finden Sie Informationen dazu, welche Versionen der Compiler und Laufzeitsysteme der einzelnen Programmiersprachen Sie zum Erzeugen vom UTM-Teilprogrammen einsetzen können. Für jede Programmiersprache, die openUTM unterstützt, sind in einer Tabelle folgende Sachverhalte aufgelistet:

- Versionen des Compilers, die Sie benutzen können, um die Objekte (OM oder LLM) eines UTM-Teilprogramms zu erzeugen
- Versionen des Laufzeitsystems, die für diese Teilprogramme geeignet sind
- Werte für den Operanden COMP der Steueranweisung PROGRAM, die Sie bei der KDCDEF-Generierung angeben müssen, um diese Teilprogramme in die Anwendungskonfiguration aufzunehmen

Den Tabellen können Sie insbesondere entnehmen, welche Kombinationen von Compiler-Version, Version des Laufzeitsystems und COMP-Wert erlaubt sind.

Weitere Informationen über zulässige Compiler- und Laufzeitsystem-Versionen finden Sie auch im Handbuch zu CRTE und in den Unterlagen zu den Compilern (Freigabemitteilung, Handbücher).

Beachten Sie, dass im Folgenden auch Compiler-Versionen aufgeführt sind, für die die Wartung bereits eingestellt ist. Der Grund dafür ist, dass in einigen Kunden-Anwendungen ältere Programme als Objekte vorhanden sind und weiterhin benützt werden sollen.

Für alle Weiter- und Neuentwicklungen sollten Sie grundsätzlich die Compiler- und Laufzeitsystem-Versionen einsetzen, die noch gewartet werden! Andernfalls haben Sie bei Problemen keinen Anspruch auf Gewährleistung oder Korrekturen.

Generell sind die Hinweise und Einschränkungen in den Freigabemitteilungen und Handbüchern der Compiler zu beachten.

> openUTM V7.0 setzt eine bestimmte CRTE-Version voraus. Details finden Sie in der Freigabemitteilung.

Sprachmix in einer Anwendung

Sprachmix bedeutet, dass eine UTM-Anwendung aus Teilprogrammen zusammengesetzt werden darf, die mit den Compilern unterschiedlicher Sprachen erzeugt wurden und die im Ablauf entsprechend unterschiedliche Laufzeitsysteme benutzen. Dabei ist jedoch Folgendes zu beachten:

Eine UTM-Anwendung darf für **jede** Programmiersprache, die von openUTM unterstützt wird und mit PROGRAM ... COMP **ungleich** ILCS generiert werden muss, nur **ein** Laufzeitsystem enthalten.

Sprachmix innerhalb eines Teilprogramms

openUTM erlaubt einen Sprachmix auch innerhalb eines einzelnen Teilprogramms, d.h. Teilprogramme können aus mehreren Quellcodes zusammengesetzt sein, die in unterschiedlichen Programmiersprachen geschrieben sind. Voraussetzung dafür ist, dass die Programmteile sich nach den gleichen Verknüpfungskonventionen aufrufen. Dabei ist anzustreben, dass sich die Programmteile nach den ILCS-Konventionen aufrufen, siehe CRTE-Handbuch. Insbesondere müssen die Datendarstellungen bei der Übergabe von Parametern und beim Zugriff auf gemeinsame Datenstrukturen identisch sein.

> **ACHTUNG!**
>
> Die Hinweise in Abschnitt „Hinweise für Anwendungen mit ILCS-Teilprogrammen“ sind unbedingt einzuhalten.

Beim Aufruf von Assembler-Unterprogrammen sind folgende Fälle zu unterscheiden:

- Sowohl das aufrufende Programm als auch das gerufene Unterprogramm sind ILCS-Programme. In diesem Fall sind keine Einschränkungen zu beachten.

  Es ist möglich, Assembler-ILCS-Programme mit Z-Makros zu programmieren. Siehe dazu openUTM-Handbuch „Anwendungen programmieren mit KDCS für Assembler“.
- Aufrufendes und gerufenes Programm sind keine ILCS-Programme, halten aber die gleichen Verknüpfungskonventionen ein. Wie diese Veknüpfungskonventionen aussehen, ist in den Benutzerhandbüchern der jeweiligen Programmiersprache beschrieben.

Durch unbeabsichtigte Sprachübergänge bei den folgenden Konstruktionen können Fehler in der Sprachverknüpfung oder Adressfehler auftreten:

- Aufruf von C-Funktionen
- Aufruf von Assembler-Funktionen mit @-Makros
- Aufruf von Fremdsoftware
- Aufruf von Datenbanken
- Aufruf von Formatierungssystemen

15.4.1 Assembler

Compiler	Laufzeitsystem	PROGRAM ...,COMP=
ASSEMBH V1.3	ASSEMBH V1.3	ILCS

15.4.2 C/C++

Compiler	Laufzeitsystem	PROGRAM ...,COMP=
C/C++ V3.2A	CRTE[1] V10.0A auf BS2000 OSD/BC V10.0 CRTE[1] V11.0A auf BS2000 OSD/BC V11.0	ILCS [2]

Anmerkungen zur Tabelle

[1] Es wird dringend empfohlen, bei Compilern und Laufzeitsystemen immer den aktuellen Korrekturstand einzusetzen.

[2]Sie können auch COMP=C setzen. Dieser Wert wird von KDCDEF aber ohne Warnung mit COMP=ILCS überschrieben.

15.4.3 Cobol

Compiler	Laufzeitsystem	PROGRAM ..., COMP=
COBOL85 V2.3A	CRTE[1] V10.0A auf BS2000 OSD/BC V10.0 CRTE[1] V11.0A auf BS2000 OSD/BC V11.0	ILCS / COB1[2]
COBOL2000 ab V1.5A	CRTE[1] V10.0A auf BS2000 OSD/BC V10.0 CRTE[1] V11.0A auf BS2000 OSD/BC V11.0	ILCS / COB1 [2]

Anmerkungen zur Tabelle

[1]Es wird dringend empfohlen, bei Compilern und Laufzeitsystemen immer den aktuellen

Korrekturstand einzusetzen.

[2] Die von COBOL85 oder COBOL2000 erzeugten Objekte sind gleichermaßen für ILCS-

Verknüpfung und Nicht-ILCS-Verknüpfung geeignet; es gibt keine Compiler-Option, um gezielt ILCS- oder Nicht-ILCS-Objekte zu erzeugen.

COMP=ILCS bedeutet, dass die ILCS-Verknüpfung verwendet wird. COMP=COB1 bedeutet Nicht-ILCS-Verknüpfung (= Verknüpfung alter Art).

Der Wert von COMP= wird nach folgenden Kriterien gesetzt:

- COMP=ILCS wird angegeben, wenn alle aufgerufenen Unterprogramme ebenfalls ILCS-fähig sind.
- COMP=COB1 muss immer dann angegeben werden, wenn der Cobol-Modul Unterprogramme aufruft, die nicht ILCS-fähig sind.

Compiler-Optionen

Die folgende Tabelle gibt einen Überblick darüber, welche COMOPT-Parameter in Abhängigkeit vom verwendeten Compiler gesetzt werden müssen und was dabei zu beachten ist.

COMOPT-Parameter	Compiler	Bemerkung
CHECK-CALLING-HIERARCHY=NO	COBOL85 COBOL2000	Nur bei COMP=COB1, d.h. bei Nicht-ILCS-Verknüpfung.
MARK-LAST-PARAMETER=YES	COBOL2000	Empfehlung

Mischen von Cobol-Programmen

Wenn ein Teilprogramm mit COMP=COB1 generiert ist, dann darf es aus Modulen bestehen, die mit dem COB1-, COBOL85- oder COBOL2000-Compiler übersetzt wurden. Wenn das Teilprogramm mit COMP=ILCS generiert ist, dann darf es nur aus Modulen bestehen, die mit dem COBOL85- oder COBOL2000-Compiler übersetzt wurden.

Innerhalb einer Anwendung können Teilprogramme, die mit COMP=ILCS generiert sind, und Teilprogramme, die mit COMP=COB1 generiert sind, nebeneinander existieren.

15.4.4 Fortran

Compiler	Laufzeitsystem	PROGRAM ...,COMP=
FOR1 V2.2C[1]	FOR1 V2.2	ILCS

Anmerkungen zur Tabelle

[1] Für den Compiler FOR1 V2.2C wird die ILCS-Fähigkeit über die Compiler-Option eingestellt:

- Wenn das Teilprogramm übersetzt wurde mit der Compiler-Option COMOPT LINKAGE=FOR1-SPECIFIC, dann muss in der PROGRAM-Anweisung COMP=FOR1 angegeben werden. Diese Compiler-Option legt fest, dass das Programm die „alte“ FOR1-Schnittstelle nutzt.
- Wenn das Teilprogramm mit der Compiler-Option COMOPT LINKAGE=STD (Standardwert) übersetzt wurde, dann muss in der PROGRAM-Anweisung COMP=ILCS angegeben werden. Diese Compiler-Option legt fest, dass das Programm die ILCS-Schnittstelle verwendet.

Bemerkung zum Sprachmix

In einer UTM-Anwendung dürfen FOR1-Teilprogramme enthalten sein, die mit unterschiedlichen Werte für PROGRAM ...,COMP= in die Konfiguration eingetragen wurden. In einer UTM-Anwendung können somit nebeneinander Teilprogramme existieren, von denen eines die „alte“ FOR1-Schnittstelle und ein anderes die ILCS-Schnittstelle nutzt.

15.4.5 Pascal

Compiler	Laufzeitsystem	PROGRAM ...,COMP=
Pascal-XT ab V2.2B	V2.2	ILCS

15.4.6 PL/I

Compiler	Laufzeitsystem	PROGRAM ...,COMP=
PLI1 V4.2	[1]	PLI1
PLI1 ab V4.2A [2]	[1]	ILCS

Anmerkungen

[1] Beachten Sie bitte, dass die Compilerobjekte die gleiche Version wie das benutzte Laufzeitsystem haben müssen. Sie dürfen nicht mehrere PLI1-Laufzeitsysteme in eine UTM-Anwendung einbinden.

[2] Mit PLI1 einer Version >= V4.2A können auch ILCS-Teilprogramme erzeugt werden.

Dazu müssen Sie bei der PROCEDURE-Anweisung OPTIONS(ILCS) angeben. Sie müssen dann darauf achten, dass Ihre UTM-Anwendung nur PL/I-Teilprogramme enthält, die ILCS-fähig sind.

15.4.7 SPL4

Compiler	Laufzeitsystem	PROGRAM ...,COMP=
SPL4 ab V2.9A	[1]	SPL4 / ILCS [2]

Anmerkungen zur Tabelle

[1] Mit openUTM wird das Laufzeitsystem SPL V2.9A ausgeliefert. Dies kann sich jedoch mit einer Korrekturausgabe ändern. Lesen Sie dazu bitte die aktuelle Freigabemitteilung.

[2]Die ILCS-Fähigkeit des Teilprogramms hängt von der Compiler-Option ab:

- Wird das Teilprogramm mit der Compiler-Option GEN=(ILCS=NO) übersetzt (Nicht-ILCS-Verknüpfung), dann muss es mit COMP=SPL4 generiert werden.
- Wird das Teilprogramm mit der Compiler-Option GEN=(ILCS=YES) übersetzt (ILCS-Verknüpfung), dann muss es mit COMP=ILCS generiert werden.

15.4.8 Hinweise für die Umstellung von einer älteren openUTM-Version

Um zu klären, welcher Wert COMP=... für ein Teilprogramm geeignet ist, prüfen Sie bitte Folgendes:

- Mit welchem Compiler ist das Teilprogramm übersetzt worden? Welche Version des Compilers wurde verwendet? Welche Compiler-Optionen wurden dabei gesetzt? Ist demnach ein ILCS-fähiges Objekt entstanden?

  Zur Klärung lesen Sie bitte die Information zu den einzelnen Compilern in den Abschnitten ab ''Assembler''.
- Ruft das Teilprogramm Unterprogramme auf? Sind diese Unterprogramme ILCS-fähig?

  Wenn sowohl das Teilprogramm als auch alle eventuellen Unterprogramme ILCS-fähig sind, dann sollte das Teilprogramm mit COMP=ILCS generiert werden.

  Wenn das Teilprogramm oder eines der Unterprogramme nicht ILCS-fähig ist oder wenn sich dies nicht klären lässt, dann gehen Sie wie folgt vor:
  - Bei Nicht-Cobol-Teilprogrammen behalten Sie zunächst den bisherigen Wert COMP=... bei.
  - Bei Cobol-Teilprogrammen setzen Sie COMP=COB1, auch dann, wenn das Programm mit dem Compiler COBOL85 übersetzt wurde. Beim Binden des UTM-Anwendungsprogramms müssen Sie das bisherige Cobol-RTS bereitstellen.

> **i** Es wird daher **dringend empfohlen**, bei einer Umstellung einer älteren UTM-Anwendung auf die ILCS-Verknüpfung überzugehen. Die ILCS-Verknüpfung hat den Vorteil, dass man einerseits Programme verschiedener Programmiersprachen problemlos untereinander aufrufen kann und dass andererseits die Wartung und Weiterentwicklung der Compiler und Laufzeitsysteme gesichert ist.

15.5 Aufbau der Accountingsätze von openUTM

Die Accountingsätze von openUTM werden in die BS2000 Accounting-Datei geschrieben und mit RAV ausgewertet. Die Sätze entsprechen den Konventionen für BS2000-Accounting-Sätze. Deshalb sind die ersten 24 Byte für das BS2000-Accounting reserviert, die restlichen Daten enthalten UTM-spezifische Informationen.

Es gibt folgende zwei Satztypen:

- Abrechnungssätze (Satztyp UTMA)
- Kalkulationssätze (Satztyp UTMK)

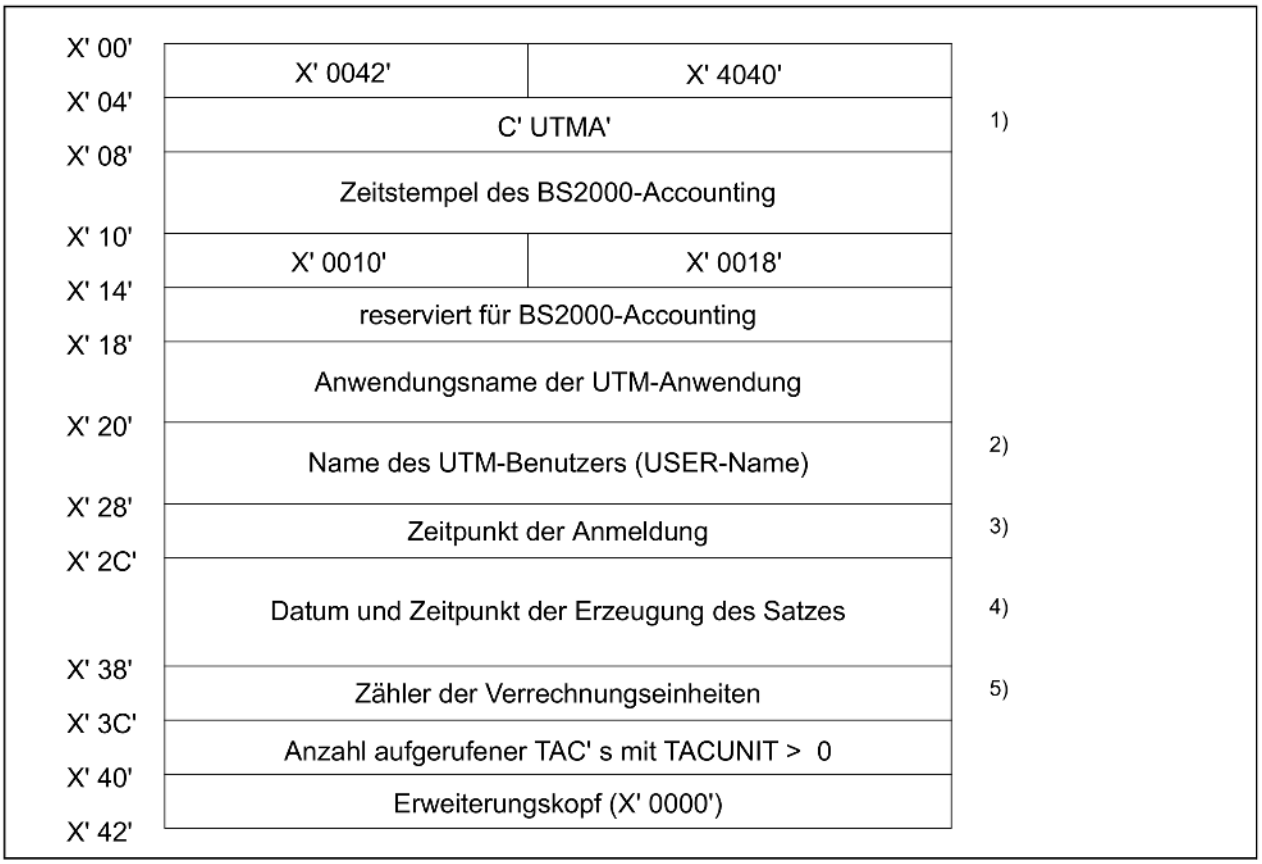
Die Bedeutung dieser Sätze ist im Kapitel „Accounting“ beschrieben.

Zum Aufbau von Abrechnungs- und Kalkulationssatz in der BS2000-Abrechnungsdatei benötigen Sie das BS2000-Makro ARDS.

Wird das ARDS mit UTM=NEW aufgerufen, dann greift ARDS auf die UTM- Makros KDCUTMA und KDCUTMK zu. KDCUTMA und KDCUTMK stehen in der UTM-Makrobibliothek SYSLIB.UTM.070.ASS. Sie beschreiben die Struktur von Abrechnungssatz bzw. Kalkulationssatz.

Wird ARDS mit UTM=OLD aufgerufen, dann werden die BS2000-eigenen Makros ARDSUTMA und ARDSUTMK verwendet.

15.5.1 Aufbau des Abrechnungssatzes



Die Anmerkungen bedeuten:

1)Kennung des Abrechnungssatzes beim BS2000-Accounting

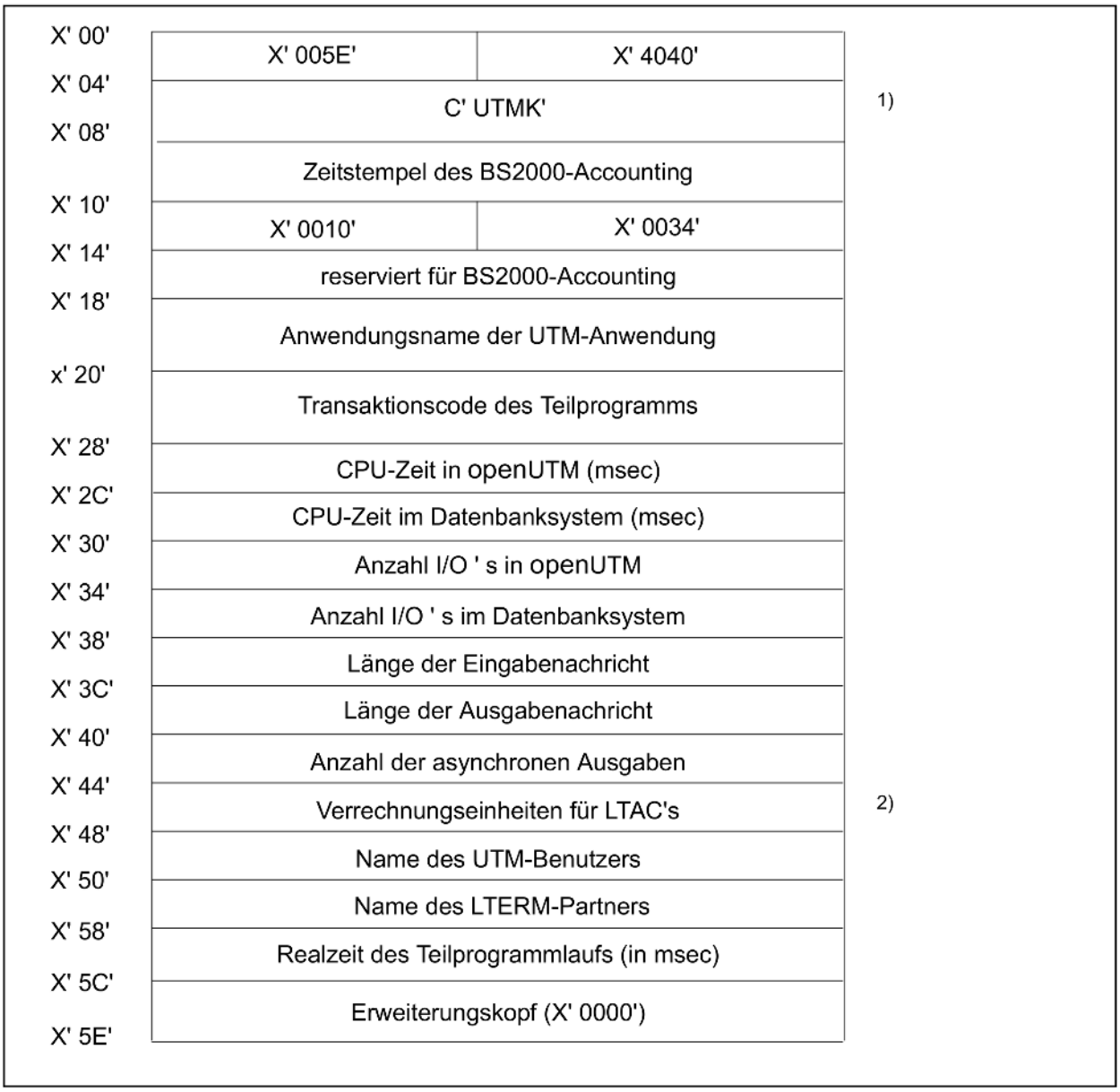
2)Name des Benutzers. In einer UTM-Anwendung ohne generierte Benutzer trägt openUTM den Namen des LTERM-Partners ein.

3)Zeitpunkt der Anmeldung dieses Benutzers (USER) an diesem LTERM in Sekunden,bezogen auf die Zeitbasis der BS2000-Version.
Wenn im aktuellen Lauf der UTM-Anwendung für diesen USER nur Asynchron-TACs aufgerufen wurden, ist der Inhalt dieses Feldes binär Null.

4)Format: *yymmddhhmmss* (Jahr/Monat/Tag/Stunde/Minute/Sekunde)

5)Summe der Verrechnungseinheiten für diesen Benutzer, seit der letzte Verrechnungssatz geschrieben wurde bzw. seit dem Zeitpunkt der Anmeldung.

15.5.2 Aufbau des Kalkulationssatzes



Die Anmerkungen bedeuten:

[1)]Kennung des Satzes beim BS2000-Accounting

[2)]siehe KDCDEF-Anweisung LTAC...,LTACUNIT=

15.6 Aufbau der SAT-Protokollsätze

Sicherheitsrelevante UTM-Ereignisse können mit der BS2000-Funktion SAT (**S**ecurity **A**udit **T**rail) protokolliert werden. Die Protokollierung von UTM-Ereignissen durch SAT ermöglicht die Beweissicherung. Bei eingeschalteter SAT-Protokollierung wird eine Mindestprotokollierung durchgeführt. Zusätzlich können weitere Ereignisse definiert werden. Die Protokollierung dieser Ereignisse in den SAT-Protokollsätzen kann Ereignis-spezifisch, Benutzer-spezifisch und Auftrags-spezifisch ein- und ausgeschaltet werden.

15.6.1 Bedeutung der von openUTM benutzten Protokolldatenfelder

Für jedes Ereignis erzeugt openUTM einen SAT-Protokollsatz. Jeder Protokollsatz, den openUTM an SAT übergibt, besteht aus einem Teil fester Struktur und Länge, dem SAT-Header, dem ein Teil variabler Struktur und Länge folgt.

Der SAT-Header enthält Datum und Uhrzeit, den BS2000-Benutzer, die TSN, das aktuelle BS2000-Ereignis und dessen Resultat. Dabei werden nur die folgenden Felder von openUTM versorgt:

Feldname	Bedeutung	Typ
EVT	Typ des Ereignisses	C-string(1..3) bei openUTM immer „TRM“
RES	Ergebnis des Ereignisses	Set: Success/Failure („S“ / „F“)

Der variable Teil enthält einzelne UTM-spezifische Datenfelder, denen jeweils ein Längenfeld und ein SAT-Identifier vorangeht. Die Art und Anzahl der einzelnen Datenfelder hängt vom Typ des UTM-Ereignisses ab.

Die folgenden Tabellen enthalten die von openUTM benutzten Protokolldatenfelder (alphabetisch sortiert), die Bedeutung des Feldinhalts und dessen Datentyp.

UTM-Ereignisse können mit der ALARM-Funktion von SAT gekoppelt werden. Der Datentyp für SAT-ALARM stimmt bis auf wenige Ausnahmen mit dem Datentyp für SATUT überein. In der folgenden Tabelle ist der ALARM-Datentyp in Klammern () angegeben, wenn er sich vom Datentyp für SATUT unterscheidet.

Feldname	Bedeutung	Typ
ACCTYP	Zugriffsart UTM-Speicherbereich	Set: C / D / READ / WRITE
APPLNAM	BCAM-Anwendungsname	C-String (1..8)
CALLER	Adresse des Aufrufers	X-String (1..4) (ALARM: X-String 4..4)
COMMAND	Name UTM-SAT-Administrationskommando oder Administrations-Programmschnittstelle	C-String (1..8)
DATNAM1	Name UTM-Speicherbereich	C-String (1..8)
DATNAM2	Name UTM-Objekt	C-String (1..8)
DATTYP	Typ des UTM-Speicherbereichs	Set: G / T / U
LTERM	LTERM-Partnername für Clients und Drucker	C-String (1..8)
MUXLTRM	MUX-Transportverbindung	C-String (1..8)
OBJECT1	Name UTM-Objekt oder Administrations-Programmschnittstelle: Name des modifizierten Feldes	C-String (1..8)

OBJECT2	Name UTM-Objekt oder Administrations-Programmschnittstelle: Inhalt des modifizierten Feldes	C-String (1..8)
OBJECT3	Administrations-Programmschnittstelle: Objekttyp	C-String (1..8)
PTERM	BCAM-Name	C-String (1..8)
TACIDEN	Transaktionskennzeichen	Set: G/T/D/C/P
TACNAM	Transaktionscode	C-String (1..8)
USER2	UTM-Benutzername	C-String (1..8)
UTMAPPL	UTM-Anwendungsname	C-String (1..8)
UTMHEX3	RC der Administrations-Programmschnittstelle	C-String (1..8)
UTMNAME	Name Lademodul oder Administrations-Programmschnittstelle: Name UTM-Objekt	C-String (1..32)
UTMOBJ4	Programm-Lademodus oder Administrations-Programmschnittstelle: Subopcode1 oder „FORMATTR“ oder Feldname	C-String (1..8) (ALARM: C-String 8..8)
UTMOBJ5	Administrations-Programmschnittstelle: Format-Attribut oder Feldname	C-String (1..8)
UTMOBJ6	Lademodul-Version oder langer Prozessorname	C-String (1..24) (ALARM: C-String 64..64)
UTMREAS	KDCS-Returncode [1] Administrations-Programmschnittstelle: leer	C-String (1..8)
UTMSTAT	Transaktionsstatus	C-String (1)
UTMSUBC	UTM-Ereignis	Set: CHANGE-PW / SIGN / DATA-ACCESS / ADM-CMD / START-PU/ END-PU / TASK-ON/ TASK-OFF / SEL-CMD/ CHG-PROG
UTMTAID	Transaktions-Identifikation [2]	X-String (1..4) (ALARM: X-String 4..4)
UTMUSER	UTM-Benutzername	C-String (1..8)

[1]Der Returncode setzt sich bei den UTM-Ereignissen CHANGE-PW, START-PU, END-PU und DATA-ACCESS zusammen aus dem kompatiblen und dem inkompatiblen KDCS-Returncode.

[2]Die Transaktions-Identifikation (TA-ID) setzt sich zusammen aus zwei Byte Vorgangszähler (innerhalb einer Session) und zwei Byte Transaktionszähler (innerhalb eines Vorgangs). Sie wird an der Schnittstelle IUTMDB zur Datenbank für die SAT-Protokollierung der Datenbank-Ereignisse bereitgestellt. Die Transaktions-Identifikation dient dazu, ein Ereignis der Transaktion zuzuordnen, die es erzeugt hat.

15.6.2 Versorgung der Datenfelder

In diesem Abschnitt wird beschrieben, welche Protokolldatenfelder bei der Protokollierung der einzelnen Ereignisse versorgt werden.

Die folgenden Tabellen geben einen Überblick darüber, welche Protokolldatenfelder in Abhängigkeit von den Ereignissen versorgt werden. Die Protokolldatenfelder sind in der Reihenfolge aufgeführt, in der sie im SAT-Protokollsatz erscheinen. Danach wird die Bedeutung der Feldinhalte für die einzelnen Ereignisse aufgeführt.

Bedeutung der Einträge in den folgenden Tabellen:

—	Feld wird nicht versorgt
Y	Feld wird versorgt (Mandatory-Felder im Sinne von SAT)
O	Feld wird in bestimmten Fällen versorgt

Feldname	**UTM-Ereignisse**									
	TASK-ON	**TASK-OFF**	**SIGN**	**CHANGE-PW**	**START-PU**	**END-PU**	**DATA-ACCESS**	**ADM-CMD**	**SEL-CMD**	**CHG-PROG**
APPLNAM	-	-	O	-	-	-	-	O	-	-
UTMUSER	-	-	Y	Y	Y	Y	Y	Y	Y	-
UTMAPPL	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UTMSUBC	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
LTERM	-	-	Y	-	O	Y	-	O	-	-
PTERM	-	-	Y	-	-	-	-	O	-	-
MUXLTRM	-	-	O	-	-	-	-	O	-	-
DATNAM1	-	-	-	-	-	-	Y	-	-	-
DATNAM2	-	-	-	-	-	-	O	O	O	-
DATTYP	-	-	-	-	-	-	Y	-	-	-
ACCTYP	-	-	-	-	-	-	Y	-	-	-
COMMAND	-	-	-	-	-	-	-	Y	Y	-
OBJECT1	Y	Y	Y	Y	Y	Y	-	O	O	-
OBJECT2	-	-	-	-	O	-	-	O	O	-
OBJECT3	-	-	-	-	-	-	-	O	-	-
CALLER	-	-	-	-	-	-	-	-	-	-
TACNAM	-	-	-	Y	O	Y	Y	Y	Y	-
TACIDEN	-	-	-	-	Y	Y	-	-	-	-

USER2	-	-	-	-	-	-	-	O	O	-
UTMNAME	-	-	-	-	-	-	-	O	-	Y
UTMTAID	-	-	Y	Y	Y	Y	Y	Y	Y	Y
UTMSTAT	-	-	-	-	-	Y	-	-	-	-
UTMREAS	-	-	Y	Y	Y	Y	Y	Y	Y	-
UTMOBJ4	-	-	-	-	-	-	-	O	-	Y
UMTOBJ5	-	-	-	-	-	-	-	O	-	-
UTMOBJ6	-	-	Y	-	-	-	-	O	-	Y
UTMHEX3	-	-	-	-	-	-	-	Y	-	-

Im Folgenden ist der Aufbau des variablen Teils des Protokollsatzes für jedes Ereignis detailliert beschrieben.

TASK-ON: Anschließen einer Task an die UTM-Anwendung

UTMAPPL	Name der laufenden Anwendung
UTMSUBC	TASK-ON
OBJECT1	F für die erste (first) Task oder N für eine Folgetask (next Task) oder L wird beim Programmaustausch oder beim PEND ER ausgelöst (Load Program)

TASK-OFF: Abmelden einer Task von der UTM-Anwendung

Es wird nur die normale Beendigung einer Task protokolliert.

UTMAPPL	Name der laufenden Anwendung
UTMSUBC	TASK-OFF
OBJECT1	Letzte Task: ja "Y" oder nein "N"

SIGN: Anmelden eines UTM-Benutzers

APPLNAM	BCAM-Anwendungsname
UTMUSER	Name des Benutzers/Clients, der das Ereignis auslöst
UTMAPPL	Name der laufenden Anwendung
UTMSUBC	SIGN
LTERM	Name des LTERM-Partners, über den sich Benutzer/Clients anschließen
PTERM	BCAM-Name der Benutzer/Clients, die definierten LTERM-Partnern zugeordnet sind

MUXLTRM	PTERM-Name der MUX-Transportverbindung
UTMTAID	Transaktions-Identifikation oder null
OBJECT1	BCAM-Prozessorname
UTMREAS	Rückgabefeld KCRSIGN
UTMOBJ6	Langer Prozessorname

CHANGE-PW: Passwortänderung

CHANGE-PW wird auch ausgelöst, wenn ein Benutzerpasswort vom UTM-Administrator geändert wird.

UTMUSER	Name des UTM-Benutzers, der das Ereignis auslöste (eventuell Administratorkennung)
UTMAPPL	Name der laufenden Anwendung
UTMSUBC	CHANGE-PW
OBJECT1	Name der Benutzerkennung, deren Passwort geändert wird
UTMREAS	KDCS-Returncodes
TACNAM	Transaktionscode des laufenden Teilprogramms
UTMTAID	Transaktions-Identifikation oder null

START-PU: Erzeugen eines Auftrags bzw. Start eines Teilprogramms

UTMUSER	Name des UTM-Benutzers, der das Ereignis auslöst
UTMAPPL	Name der laufenden Anwendung
UTMSUBC	START-PU (start program unit)
LTERM	Name des definierten LTERM-Partners oder leer
OBJECT1	Bei TACIDEN=G: TAC des Auftrags Bei TACIDEN=C/T/P: TAC des laufenden Vorgangs
OBJECT2	Bei TACIDEN=G: DPUT-Identifikation des erzeugten Auftrags
UTMREAS	KDCS-Returncodes (bei TACIDEN=G)
UTMTAID	Transaktions-Identifikation der laufenden Transaktion oder null
TACNAM	TAC des laufenden Teilprogramms
TACIDEN	Transaktionskennzeichen. Mögliche Werte (die Werte G, C, T, P schließen sich dabei gegenseitig aus):

	G für Auftrag erzeugt (generate) Dialog-Aufträge werden nur als erzeugt protokolliert, wenn sie wegen TAC-Klassensteuerung nicht sofort gestartet werden können
	C für Vorgangsbeginn (begin conversation) Mit diesem Teilprogramm beginnt ein Vorgang
	T für Transaktionsbeginn (begin transaction) Mit diesem Teilprogramm beginnt eine Folgetransaktion eines Vorgangs
	P für Beginn eines Folgeteilprogramms innerhalb einer Transaktion (begin program run)

Das Erzeugen von Nachrichten an einen LTERM-Partner wird nicht protokolliert.
Ein Quittungsauftrag (an einen TAC) wird erst dann als erzeugt protokolliert, wenn er auf Grund des Ergebnisses des Hauptauftragslaufes ausgewählt und in einen Hauptauftrag umgewandelt wird. Das Ziel des abgelaufenen Hauptauftrags wird dabei im Feld LTERM bzw. TACNAM ausgegeben.

END-PU: Beendigung eines Teilprogramms

UTMUSER	Name des UTM-Benutzers, der das Ereignis auslöst
UTMAPPL	Name der laufenden Anwendung

UTMSUBC	END-PU (end program unit)
LTERM	Name des definierten LTERM-Partners oder leer
OBJECT1	Transaktionscode des laufenden Vorgangs
UTMSTAT	Das Feld wird nur bei TACIDEN=T oder C versorgt. Es enthält dann den Transaktionsstatus:
	C Transaktionssicherung (Commit)
	R Rücksetzen der Transaktion (Rollback)
UTMTAID	Transaktions-Identifikation
TACNAM	Transaktionscode des laufenden Teilprogramms
TACIDEN	Mögliche Werte:
	C Ende des Teilprogramms und des Vorgangs (end of conversation)

	T Ende des Teilprogramms und der Transaktion; der Vorgang wird fortgesetzt (end of transaction)
	P Ende des Teilprogramms; die Transaktion wird fortgesetzt (end of program run)
UTMREAS	KDCS-Returncodes

DATA-ACCESS: Zugriff auf einen UTM-Speicherbereich

UTMUSER	Name des UTM-Benutzers, der das Ereignis auslöst
UTMAPPL	Name der laufenden Anwendung
UTMSUBC	DATA-ACCESS
DATNAM1	Name des angesprochenen UTM-Speicherbereiches
DATNAM2	Bei DATTYP=ULS: UTM-Benutzer, bei DATTYP=TLS: LTERM-Partner für Clients und Drucker sonst leer
DATTYP	Type des Speicherbereichs:
	G: GSSB
	U: ULS
	T: TLS
ACCTYP	Art des Speicherzugriffs:
READ	lesend
WRITE	schreibend
	C: erzeugt (create)
	D: gelöscht (delete)
UTMREAS	KDCS-Returncodes
UTMTAID	Transaktions-Identifikation
TACNAM	Transaktionscode des laufenden Teilprogramms

ADM-CMD: Aufruf der Administrations-Programmschnittstelle

UTMUSER	Name des UTM-Benutzers, der das Ereignis auslöst

UTMAPPL	Name der laufenden Anwendung
UTMSUBC	ADM-CMD
COMMAND	DADM (KDCS-Aufruf) oder Aufruf der Programmschnittstelle zur Administration:
	CHNGAPPL / CREATE / CREASTMT / DELETE / DUMP / ENCRYPT
	/GETOBJ / LOCKMGMT / MODIFY / ONLIMP / PETTA / SENDMSG
	/SHUTDOWN / SPOOLOUT / SYSLOG / UPDIPADR / USLOG
UTMHEX3	Returncode der Administrations-Programmschnittstelle
UTMTAID	Transaktions-Identifikation
USER2	Session, Benutzer oder leer

Die Felder DATNAM2, LTERM und OBJECT1 werden eventuell leer ausgegeben.

> Bei manchen Funktionsaufrufen der Programmschnittstelle sind verschiedene Aktionen möglich. In diesem Fall schreibt SAT für jede Aktion einen Protokollsatz und protokolliert den Parameter, wobei nur für geänderte Parameter auch der Parameterwert ausgegeben wird.
> Sehen Sie auch openUTM-Handbuch „Anwendungen administrieren“.

In Abhängigkeit von COMMAND werden in bestimmten Fällen zusätzlich folgende Felder versorgt:

- COMMAND: CHNGAPPL
  UTMOBJ4: Subopcode1 (NEW/OLD bei PROGRAM)
- COMMAND: CREATE
  OBJEC:T3: Objekt-Typ
  UTMNAME: Objekt-Name
- COMMAND: DELETE
  OBJECT3: Objekt-Typ
  UTMOBJ4: Subopcode1 (DELAY/IMMEDIAT)
  UTMNAME: Objekt-Name
- COMMAND: ENCRYPT
  OBJECT1: Subopcode1 (CREATEK, ACTIVATK, DELETEK, REAACTK, REANEWK)
- COMMAND: LOCKMGMT
  OBJECT1: Subopcode1
- COMMAND: MODIFY
  OBJECT3: Objekt-Typ bzw. Parameter-Typ
  In Abhängigkeit des Objekt-Typs bzw. Parameter-Typs werden weitere Felder protokolliert:

Objekt-Typ	**protokollierte Felder**

CLNODE	OBJECT1: Parameter UTMOBJ6: Parameterwert
CON	UTMOBJ6: Langer Prozessorname
KSET	OBJECT1: Keys
LOADMODU	OBJECT1: „Version“ UTMOBJ6: Version UTMNAME: Name des zu modifizierenden Lademoduls
LPAP	OBJECT1: Parameter OBJECT2: Parameterwert UTMNAME: Name des zu modifizierenden LPAP-Partners
LSES	OBJECT1: Parameter OBJECT2: Parameterwert UTMNAME: Name des zu modifizierenden LSES- und CON-Tripels UTMOBJ6: Langer Prozessorname
LTAC	OBJECT1: Parameter OBJECT2: Parameterwert UTMNAME: Name des zu modifizierenden LTACs
LTERM	OBJECT1: Parameter OBJECT2: Parameterwert UTMNAME: Name des zu modifizierenden LTERM-Partners UTMOBJ4: „FORMATATTR“ UTMOBJ5: Formatattribut
MUX	OBJECT1: Parameter OBJECT2: Parameterwert UTMNAME: Name des zu modifizierenden MUX-Tripels
OSICON	OBJECT1: Parameter OBJECT2: Parameterwert UTMNAME: Name des zu modifizierenden OSI-CON
OSILPAP	OBJECT1: Parameter OBJECT2: Parameterwert UTMNAME: Name des zu modifizierenden OSI-LPAP-Partners
PTERM	PTERM: Name des zu modifizierenden Clients/Drucker (PRTM) OBJECT1: Parameter OBJECT2: Parameterwert UTMNAME: Name des zu modifizierenden PTERM-Tripels UTMOBJ6: Langer Prozessorname

TAC	OBJECT1: Parameter OBJECT2: Parameterwert UTMNAME: Name des zu modifizierenden TACs UTMOBJ6: Parameterwert
TACCLASS	OBJECT1: Parameter OBJECT2: Parameterwert UTMNAME: Name der zu modifizierenden TACCLASS
TPOOL	LTERM: LTERM-Präfix OBJECT1: Parameter1 OBJECT2: Parameterwert1 UTMOBJ4: Parameter2 UTMOBJ5: Parameterwert2 UTMNAME: Name des zu modifizierenden TPOOLs (LTERM-Präfix, PRONAM, PTYPE, BCAMAPPL)
USER	OBJECT1: Parameter OBJECT2: Parameterwert UTMOBJ4: „FORMATTR“ UTMOBJ5: Formatattribut UTMNAME: Name des zu modifizierenden USERs

Parameter-Typ	**protokollierte Felder**
CCURRPAR	OBJECT1: Parameter UTMOBJ4: Parameterwert 1 UTMOBJ5: Parameterwert 2
CLPAR	OBJECT1: Parameter UTMOBJ4: Parameterwert
CURRPAR	OBJECT1: Parameter UTMOBJ6: Parameterwert
DIAGACCP	OBJECT1: Parameter OBJECT2: Parameterwert 1 UTMOBJ6: Parameterwert 2
MAXPAR	OBJECT1: Parameter OBJECT2: Parameterwert
TASKSPAR	OBJECT1: Parameter OBJECT2: Parameterwert
TIMERPAR	OBJECT1: Parameter OBJECT2: Parameterwert

- COMMAND: ONLIMP

  OBJECT1: „KC_ALL“

- COMMAND: SENDMSG

  LTERM: LTERM-Name oder „KDCALL“
- COMMAND: SHUTDOWN

  OBJECT1: Subopcode1

  OBJECT2: Parameterwert
- COMMAND: SPOOLOUT

  OBJECT1: „SPOOLOUT“

  OBJECT2: „ON“
- COMMAND: SYSLOG

  OBJECT1: Subopcode1

  UTMOBJ6: Parameterwert
- COMMAND: UPDIPADR

  OBJECT1: Subopcode1
- COMMAND: USLOG

  OBJECT1: Subopcode1

SEL-CMD: Ausführung eines Preselection-Kommandos

Ist der Name oder Wert eines Parameters länger als 8 Zeichen, so wird die bei der UTM-SAT-Administration beschriebene Abkürzung im Protokolldatenfeld ausgegeben.

- UTMUSER: Name des UTM-Benutzers, der das Ereignis auslöst
- UTMAPPL: Name der laufenden Anwendung
- UTMSUBC: SEL-CMD
- COMMAND: Angabe für das UTM-SAT-Administrationskommando: MSATSEL oder MSATPROT
- UTMREAS: Systeminterner Returncode
- UTMTAID: Transaktions-Identifikation
- TACNAM: Transaktionscode des laufenden Teilprogramms
- USER2: Benutzer oder leer

Die Felder DATNAM2, LTERM und OBJECT1 werden eventuell leer ausgegeben.

In Abhängigkeit von COMMAND werden zusätzlich folgende Felder versorgt:

- MSATSEL (Preselection steuern, Kommando KDCMSAT SATSEL=...)

  Für jeden angegebenen Namen wird ein eigener Protokolldatensatz geschrieben. Es wird immer nur genau eines der Felder USER2, DATNAM2 oder OBJECT1 versorgt:

  USER2: UTM-Benutzer als Objekt zur Preselection

  DATNAM2: TAC als Objekt zur Preselection

  OBJECT1: Ereignis als Objekt zur Preselection

  OBJECT2: Preselection-Wert (NONE, SUCC, FAIL, BOTH oder OFF)
- MSATPROT (SAT-Protokollierung steuern, Kommando KDCMSAT SAT=...)

  OBJECT2: Protokollierung ein- (ON) bzw. ausgeschaltet (OFF)

CHG-PROG: Austausch eines Lademoduls

- UTMAPPL: Name der laufenden UTM-Anwendung
- UTMSUBC: CHG-PROG
- UTMNAME: Name des auszutauschenden Moduls
- UTMOBJ4: Lade-Modus des auszutauschenden Moduls
- UTMOBJ6: Neue Modulversion
- UTMTAID: Transaktions-Identifikation, falls UTMOBJ4=ON-CALL, sonst null

> ℹ Für UTMOBJ4=ON-CALL ist zu beachten, dass CHG-PROG immer als Erfolg protokolliert wird. Ob der Austausch erfolgreich war, kann am zugehörigen END-PU-Protokolldatensatz abgelesen werden. (Im Fehlerfall kommt ein PEND ER mit entsprechenden KDCS-Returncodes.) Auch das erstmalige Laden wird als Austausch protokolliert.

15.7 Beispielprogramme

Für openUTM werden standardmäßig Beispielprogramme ausgeliefert, die dem Benutzer Teilaspekte der Anwendungserstellung erleichtern sollen. Nachfolgend werden einige dieser Beispielprogramme näher erläutert.
Die Beschreibung der Beispielprogramme für die Administration finden Sie im openUTM-Handbuch „Anwendungen administrieren“.

15.7.1 Beispielprogramme zum Anmelde-Vorgang

Mit den Beispielprogrammen zum Anmelde-Vorgang kann ein einfacher Anmelde-Vorgang mit FHS-Formatausgabe realisiert werden.

Es werden sowohl übersetzte Objekte als auch Quellprogramme (Cobol) ausgeliefert. Sie haben damit die Möglichkeit, die neuen Funktionen ohne Programmierung schnell auszuprobieren. Mit den Quellprogrammen haben Sie eine Programmiervorlage für den Fall, dass Sie einen Anmelde-Vorgang nach eigenen Wünschen realisieren möchten.

Funktionen

Die Programme sind für alle Generierungsvarianten geeignet.

Der Terminalbenutzer wird nach Verbindungsaufbau (wenn erforderlich) zur Eingabe seiner Berechtigungsdaten aufgefordert. Zu diesem Zweck gibt der Anmelde-Vorgang ein „Begrüßungsformat“ aus, das zwei Eingabefelder für Benutzerkennung und Passwort enthält. Das Passwort ist abdruckbar anzugeben. Bitte beachten Sie bei der Angabe von Benutzerkennung und Passwort, dass Kleinbuchstaben in Großbuchstaben umgesetzt werden.

Wenn openUTM die Berechtigungsdaten ablehnt, dann wiederholt der Anmelde-Vorgang die Eingabe-Aufforderung. Er gibt dazu das gleiche Format aus, es enthält nun einen Hinweis auf die Ablehnung. Nach drei Fehlversuchen wird der Anmelde-Vorgang abgebrochen.

Akzeptiert openUTM die Berechtigungsdaten, verhält sich der Anmelde-Vorgang wie folgt:

- Kein Vorgangswiederanlauf:
  Ist ein Startformat generiert, so wird es ausgegeben, andernfalls eine Aufforderung zur Eingabe im Zeilenmodus.
- Vorgangswiederanlauf:
  Der Bildschirmwiederanlauf des offenen Vorgangs wird veranlasst.

Am Terminal werden englische Texte ausgegeben. Auch die Kommentare im Quellprogramm sind englisch.

Bestandteile

Die Beispielprogramme zum Anmelde-Vorgang finden Sie in der Bibliothek SYSLIB.UTM.070.EXAMPLE.

Element	LMS-Typ	Bedeutung
SIGN1	S	Cobol-Source --> 1. Teilprogramm
SIGN2	S	Cobol-Source --> 2. Teilprogramm
KDCSIGN1	R	Bindemodul --> 1. Teilprogramm
KDCSIGN2	R	Bindemodul --> 2. Teilprogramm
FORSIGN	R	Bildschirmformat (' *' -Format)
FORSIGN	S	Adressierungshilfe
FORSIGN	F	IFG-Source

Integration in eine UTM-Anwendung

Um den Beispiel-Anmelde-Vorgang in eine UTM-Anwendung zu integrieren, sind die KDCDEF-Generierungsanweisungen wie folgt zu erweitern:

```
PROGRAM KDCSIGN1,COMP=ILCS
PROGRAM KDCSIGN2,COMP=ILCS
TAC   KDCSGNTC,PROGRAM=KDCSIGN1
TAC   TACSIGN2,PROGRAM=KDCSIGN2,CALL=NEXT
```

Der TAC-Name TACSIGN2 ist einprogrammiert. Er ist am Anfang der beiden Teilprogramme jeweils als Konstante definiert und dadurch bei Bedarf leicht änderbar.

Der SPAB muss mindestens 600 Byte, der KB-Programmbereich mindestens 2 Byte lang sein (siehe MAX-Anweisung im openUTM-Handbuch „Anwendungen generieren“).

Die Binder-Anweisung ist wie folgt zu erweitern:

```
INCLUDE-MODULES LIBRARY=$userid.SYSLIB.UTM.070.EXAMPLE ,ELEMENT=(KDCSIGN1, KDCSIGN2)
```

Das Bildschirmformat FORSIGN ist in die Formatbibliothek der Anwendung aufzunehmen. Das Format ist für Terminals des Typs 8160, 9750, 9755 und 9763 geeignet.

Das Teilprogramm SIGN1 verwendet die COBOL85-spezifische Anweisung EVALUATE.

15.7.2 Beispielprogramme für Publish / Subscribe Server

Mit diesen Beispielprogrammen soll gezeigt werden, wie ein einfacher Publish- und Subscribe-Dienst in einer UTM-Anwendung realisiert werden kann.

Funktion

Ein Benutzer kann sich beim Dienst anmelden (subsrcibe). Er bekommt dann alle ab diesem Zeitpunkt veröffentlichten Nachrichten (publish) in seiner USER-Queue zugestellt. Die möglichen Kommandos an den Dienst sind:

- help: Hilfetext holen
- subscribe: Nachrichten abonnieren
- unsubscribe: Nachrichten abbestellen
- who: Namen der Abonnenten ausgeben
- publish <*message*>: Nachricht veröffentlichen

Der Dienst wird von einem Asynchron-Vorgang mit dem TAC PUBSUBA erbracht, der ständig an der TAC-Queue PUBSUBMQ auf Aufträge wartet. Die Benutzer kommunizieren mit dem Dienst über den Dialog-Vorgang PUPSUBD. Auftragsbestätigungen werden an die USER-Queue des Benutzers gesendet und können z.B. mit dem Dialog-Programm UPDGET gelesen werden (siehe Beispielprogramme zur Asynchron-Verarbeitung für UPIC-Client). Außerdem kann in jedem Teilprogramm beim INIT PU abgefragt werden, ob Nachrichten in der Queue des Benutzers vorliegen.

Der Dienst muss nur einmal durch Aufruf des TAC PUBSUBA gestartet werden. Der offene Asynchron-Vorgang bleibt dann auch über den Anwendungslauf hinweg erhalten. Nach Neugenerierung wird er durch KDCUPD in die neue Anwendung übertragen.

Sollte sich durch einen Fehler der Asynchron-Vorgang abnormal beenden, so wird der zuletzt bearbeitete Auftrag in die Dead Letter Queue gestellt.

Auslieferung

In BS2000 werden Quellprogramme und Objektmodule als Elemente der Bibliothek SYSLIB.UTM.070.EXAMPLE ausgeliefert.

Element	LMS-Typ	Bedeutung
PUBSUBD.C	S	Auftragserteilung an publish/subscribe Dienst, Dialog-Teilprogramm
PUBSUBA.C	S	Realisiert publish/subscribe Dienst, Asynchron-Teilprogramm
PUBSUBD#LLM	R	Objektmodul zu PUBSUBD.C
PUBSUBA#LLM	R	Objektmodul zu PUBSUBA.C

UTM-Generierung

Die Anweisungen für die Teilprogramme im KDCDEF-Lauf sind in den einzelnen Sourcen als Kommentar angegeben. Ebenso die Anweisung für die TAC-Queue „PUBSUBMQ“.

Es muss mindestens ein GSSB generiert werden (MAX GSSBS), da der Service zur Verwaltung der Abonnenten den GSSB „PUBSUBGB“ benutzt.

Soll nach Abbruch des Service der zuletzt bearbeitete Auftrag in die Dead Letter Queue gestellt werden, so muss MAX REDELIVERY = (...,0) generiert werden. Ansonsten bleibt er in der Auftragsqueue PUBSUBMQ.

15.7.3 Beispielprogramm für selektives Verschieben aus der Dead Letter Queue

Funktion

Das Dialog-Programm verschiebt alle Nachrichten der Dead Letter Queue mit vorgegebenem ursprünglichen Ziel an ein vorgegebenes neues Ziel. Als Eingabe werden daher insgesamt 16 Zeichen erwartet, d.h. je nach Typ des ursprünglichen Ziels entweder zwei TACs oder zwei LPAP-Partner oder zwei OSI-LPAP-Partner. Das Programm gibt zur Bestätigung die Anzahl der verschobenen Nachrichten aus.

Auslieferung

Im BS2000 werden Quellprogramme und Objektmodule als Elemente der Bibliothek SYSLIB.UTM.070.EXAMPLE ausgeliefert.

Element	LMS-Typ	Bedeutung
DADMMVS	S	Verschieben von Nachrichten mit bestimmtem ursprünglichen Ziel aus der Dead Letter Queue, COBOL-Dialog-Teilprogramm
DADMMVSC.C	S	Analoges Dialog-Programm in C
DADMMVS	R	Objektmodul zu DADMMVS
DADMMVS#LLM	R	Objektmodul zu DADMMVSC.C

UTM-Generierung

Die Anweisungen für die Teilprogramme im KDCDEF-Lauf sind in den einzelnen Sourcen als Kommentar angegeben.

15.7.4 CPI-C-Beispielprogramme

CPI-C-Beispielprogramme finden Sie in der Bibliothek SYSLIB.UTM.070.XOPEN.

Element	LMS-Typ	Bedeutung
KCPSAM1.C	S	C-Source (asynchroner Teil)
KCPSAM2.C	S	C-Source (synchroner Teil)
KCPSAM1#LLM	R	Objektmodul (asynchroner Teil)
KCPSAM2#LLM	R	Objektmodul (synchroner Teil)

15.7.5 Beispielprogramme zur Asynchron-Verarbeitung für UPIC-Clients

Für die Asynchron-Verarbeitung für UPIC-Clients werden mit openUTM die drei Teilprogramme UPDIAL, UPASYN und UPDGET ausgeliefert.

Funktionen

Mit diesen drei Teilprogrammen wird gezeigt, wie man von einem UPIC-Client aus asynchrone Aufträge erteilen und asynchron über das Ergebnis informiert werden kann.

In diesem Beispiel wird die Asynchron-Nachricht zunächst an die USER-Queue der Benutzerkennung geschickt, unter der sich der UPIC-Client angemeldet hat. Anschließend wird die Nachricht von einem Dialog-Teilprogramm gelesen und am Client ausgegeben. Der Vorteil asynchroner Verarbeitung liegt z.B. darin, dass der Benutzer am UPIC-Client sofort nach der Auftragsannahme einen neuen Auftrag eingeben kann und nicht bis zur Auftragsbeendigung blockiert ist.

Diese drei Programme haben folgende Funktionen:

- UPDIAL liest eine Eingabenachricht, schickt diese als Asynchron-Auftrag an das Teilprogramm UPASYN und gibt eine Auftragsannahmebestätigung am Client aus.
- UPASYN empfängt diese Nachricht, wartet 5 Sekunden zur Simulation einer komplexen Verarbeitung und schreibt das Ergebnis in die USER-Queue der Benutzerkennung, unter der sich der UPIC-Client angemeldet hat.
- UPDGET liest mit Warten (60 Sekunden) aus der USER-Queue der Benutzerkennung des Vorgangs (bei leerer Dialog-Nachricht) bzw. der in der Dialog-Nachricht mitgegebenen Benutzerkennung. Dadurch kann der Vorgang UPDGET unter einer anderen Benutzerkennung (z.B. ohne Verwendung eines Security Users) als die Vorgänge UPDIAL und UPASYN ablaufen und die Nachricht aus der User-Queue des Benutzers abholen, der den Vorgang UPDIAL gestartet hat.

  Enthält die nicht leere Dialog-Nachricht keine gültige Benutzerkennung, beendet sich UPDGET mit einer Fehlermeldung. Falls keine Queue-Nachricht vorliegt, wird UPDGET erneut gestartet, sobald eine Nachricht eintrifft, oder die Wartezeit abgelaufen ist. Falls eine Queue-Nachricht vorliegt, wird diese mit MPUT an den UPIC-Client gesendet und mit PEND-RE der eigene TAC nochmals aufgerufen, um auf die nächste Nachricht zu warten.

Bestandteile

Die Quellprogramme und Objektmodule werden als Elemente der Bibliothek SYSLIB.UTM.070.EXAMPLE ausgeliefert. Die Beispielprogramme sind nur in Verbindung mit einem darauf abgestimmten UPIC-Clientprogramm sinnvoll einsetzbar.

Integration in eine UTM-Anwendung

Falls der Client nur mit Benutzern betrieben werden soll, die mit RESTART=NO generiert wurden, ist der Ablauf wie folgt:

Der UPIC-Client hält zwei Verbindungen zur UTM-Anwendung und meldet sich jeweils mit derselben Benutzerkennung bei openUTM an. Der Client startet im ersten Thread den Dialog-Vorgang UPDGET zum Lesen der Asynchron-Nachrichten. Im zweiten Thread startet der Client auf explizite Anforderung den Dialog-Vorgang UPDIAL, der dann einen Asynchron-Auftrag an UPASYN erzeugt.

Soll der Client auch mit Benutzern betrieben werden die mit RESTART=YES generiert wurden, so kann folgendermaßen vorgegangen werden:

Der Client meldet sich nur im zweiten Thread mit dem Security-User an, im ersten Thread aber ohne expliziten Security-User. openUTM weist ihm also einen, der Verbindung fest zugeordneten Benutzer, zu. Der erste Thread liefert dann beim Starten des Vorgangs UPDGET (und bei jedem Dialogschritt) den Namen des Security-Users in der Dialog-Nachricht mit.

15.7.6 Beispielprogramme für HTTP-Clients

In der Datei SYSLIB.UTM.070.EXAMPLE stehen vier Beispielprogramme für HTTP-Clients als Source und LLM.

Näheres zu Funktion und Generierung ist in den Sourcen zu finden. Es sind die folgenden Programme:

- HTTPECHO.C
- HTTPEXH.C
- HTTPEXT.C
- HTTPINF.C

15.8 Beispielprozeduren

Mit openUTM werden standardmäßig Beispielprozeduren ausgeliefert, die Ihnen die Arbeit mit openUTM erleichtern sollen. Die Prozeduren sind als Werkzeuge und Vorlage gedacht, die Sie nach Ihren Wünschen modifizieren und erweitern können. Die Prozeduren enthalten teilweise englische Kommentare.

Diese Prozeduren sind schon kompiliert (SYSJ-Elemente), damit sie auch im Grundausbau mit SDF-P-BASYS ablauffähig sind. Die Sourcen sind in SYSLIB.UTM.070.EXAMPLE enthalten.

Die Bibliothek SYSPRC.UTM.070 enthält folgende Prozeduren:

Prozedur	**Funktion**
BTRACE	Mischen von BCAM-Trace-Ausgabedateien und auswerten mit dem Programm KDCBTRC
DUMP	UTM-Dumps auswerten
FGGUSLOG	Benutzerprotokoll-Datei als Dateigenerationsgruppe (FGG) einrichten und auf neue Benutzer-Protokolldatei-Generation umschalten
GEN	UTM-Anwendung generieren
LINK	UTM-Anwendungsprogramm binden
MSGMOD	Benutzereigene (Meldungs-)module erzeugen
PAMSAM	Von KDCMON aufgezeichnete Daten aufbereiten und sortieren
SHOW-ETPND	ETPND eines UTM-Moduls anzeigen
SLOG-FGG	Einzelne Dateigenerationen einer SYSLOG-FGG auswerten
START-APPL-ENTER-PROC	Erzeugen einer SDF-Prozedur zum Starten einer UTM-Anwendung mittels ENTER-PROC
START-BLS-APPLICATION	UTM-Produktiv-Anwendung mit BLS starten
SYSLOG	SYSLOG-Datei aufbereiten
UPD	Daten in KDCFILE übertragen mit KDCUPD

15.9 XS-Unterstützung von UTM-Anwendungen

In diesem Abschnitt sind einige Besonderheiten aufgelistet, die Sie beim Binden und Starten von UTM-Anwendungen berücksichtigen müssen, die in den oberen Adressraum einer XS-Anlage geladen werden und im 31-Bit-Modus ablaufen sollen. Dabei werden Kenntnisse der XS-Programmierung und zum XS-Einsatz vorausgesetzt.

openUTM unterstützt nur Anwendungen, die:

- entweder vollständig XS-fähig sind und im 31-Bit-Modus ablaufen oder
- nicht XS-fähig ganz im unteren Adressraum (d.h. unterhalb 16 MByte) geladen sind und im 24-Bit-Modus ablaufen

Eine UTM-Anwendung kann also nur dann im oberen Adressraum (>16MByte) geladen werden und im 31-Bit-Modus ablaufen, wenn alle Bestandteile des UTM-Anwendungsprogramms XS-fähig sind.

> **ACHTUNG!**
>
> UTM-Anwendungen im „mixed mode“ (d.h. mit Umschaltung zwischen 24- und 31-Bit-Adressmodus) werden von openUTM nicht unterstützt. Das bedeutet, dass openUTM nicht für den ordnungsgemäßen Ablauf einer UTM-Anwendung garantiert, wenn z.B. ein Teilprogramm, das im 31-Bit-Modus abläuft, selbstständig Module im 24-Bit-Modus nachlädt und beim Übergang jeweils den Adressmodus umschaltet.

Übersetzen und Binden

Beim Übersetzen und Binden einer XS-fähigen UTM-Anwendung sind folgende Regeln zu berücksichtigen:

- Alle Teilprogramme müssen mit den Attributen AMODE=ANY (Adressierungsmodus) und RMODE = ANY (Residenzmodus) übersetzt werden.
- Beim Binden der UTM-Anwendung überprüft der Binder die Attribute AMODE und RMODE für alle Teilprogramme und legt für das erzeugte Bindemodul der UTM-Anwendung einen Pseudo-RMODE fest. Dabei orientiert er sich an der „schwächsten“ Komponente, d.h. das Modul erhält nur dann das Attribut RMODE=ANY, wenn alle Komponenten das Attribut RMODE=ANY haben. Wurde eine Komponente mit RMODE=24 übersetzt, dann wird dem Modul RMODE=24 zugeordnet.

  Das Attribut AMODE wird durch den Programmabschnitt (CSECT) bestimmt, der die Einsprungstelle des Bindemoduls enthält.
  Nähere Informationen finden Sie im BS2000-Handbuch „Bindelader-Starter“.

Besonderheiten beim Start einer UTM-Anwendung

Ob die UTM-Anwendung in den oberen oder unteren Adressraum geladen wird, ist abhängig vom UTM-Anwendungsprogramm selbst und vom Wert des Parameters PROGRAM-MODE, den Sie beim Aufruf des START-EXECUTABLE-PROGRAM-Kommandos angeben.

- Bei PROGRAM-MODE=24 wird die Anwendung in den unteren Adressraum geladen und der 24-Bit-Modus wird eingestellt.
- Bei PROGRAM-MODE=ANY:
  Ob die Anwendung in den oberen oder unteren Adressraum geladen wird und welcher Adressierungsmodus eingestellt wird, ist abhängig von den Attributen AMODE und RMODE des Lademoduls (siehe Abschnitt „XS-Unterstützung von UTM-Anwendungen“).

  Wenn das Binder-Lader-System (BLS) beim Starten der UTM-Anwendung feststellt, dass alle Bestandteile der UTM-Anwendung, die beim Start der Anwendung geladen werden, XS-fähig sind, dann wird die UTM-Anwendung in den oberen Adressraum geladen.

Soll openUTM in der Startphase einer in den oberen Adressraum geladenen Anwendung ein nicht XS-fähiges Modul nachladen, dann bricht openUTM den Startvorgang mit einer entsprechenden Fehlermeldung ab.

Zu beachten ist auch, dass von einem Anwendungsprogramm, das im oberen Adressraum läuft, im laufenden Betrieb kein 24-Bit-Modul (ONCALL) nachgeladen werden darf.

Um sicherzustellen, dass eine nicht-XS-fähige UTM-Anwendung in den unteren Adressraum geladen wird und im 24-Bit-Modus abläuft, müssen Sie beim Binden der UTM-Anwendung eine MODIFY-SYMBOL-ATTRIBUTES-Anweisung mit AMODE=24 einfügen.

Speicherbelegung von UTM-Anwendungen

Die Anwendungs-spezifischen Tabellen und Datenbereiche (KAA, KTA, Slots und UTM-Cache) legt openUTM im Klasse 5 Speicher im oberen Adressraum an. UTM-Anwendungen, die im unteren Adressraum laufen, wird dadurch kein Adressraum weggenommen. Der UTM-Cache kann per Generierung MAX CACHE auch in einen oder mehrere Datenräume gelegt werden.

Tools KDCDEF, KDCDUMP und KDCUPD

Die UTM-Tools KDCDEF, KDCDUMP und KDCUPD laufen nur im oberen Adressraum (> 16MByte) ab.

16 Fachwörter

Fachwörter, die an anderer Stelle erklärt werden, sind mit *kursiver* Schrift ausgezeichnet.

Ablaufinvariantes Programm
reentrant program

siehe *reentrant-fähiges Programm*.

Abnormale Beendigung einer UTM-Anwendung
abnormal termination of a UTM application

Beendigung einer *UTM-Anwendung*, bei der die *KDCFILE* nicht mehr aktualisiert wird. Eine abnormale Beendigung wird ausgelöst durch einen schwerwiegenden Fehler, z.B. Rechnerausfall, Fehler in der Systemsoftware. Wird die Anwendung erneut gestartet, führt openUTM einen *Warmstart* durch.

Abstrakte Syntax (OSI)
abstract syntax

Eine abstrakte Syntax ist die Menge der formal beschriebenen Datentypen, die zwischen Anwendungen über *OSI TP* ausgetauscht werden sollen. Eine abstrakte Syntax ist unabhängig von der eingesetzten Hardware und der jeweiligen Programmiersprache.

Access-List
access list

Eine Access-List definiert die Berechtigung für den Zugriff auf einen bestimmten *Service*, auf eine bestimmte *TAC-Queue* oder auf eine bestimmte *USER-Queue*. Eine Access-List ist als *Keyset* definiert und enthält einen oder mehrere *Keycodes*, die jeweils eine Rolle in der Anwendung repräsentieren. Benutzer, LTERMs oder (OSI-)LPAPs dürfen nur dann auf den Service oder die *TAC-Queue*/*USER-Queue* zugreifen, wenn ihnen die entsprechenden Rollen zugeteilt wurden, d.h. wenn ihr *Keyset* und die Access-List mindestens einen gemeinsamen *Keycode* enthalten.

Access Point (OSI)

siehe *Dienstzugriffspunkt.*

ACID-Eigenschaften
ACID properties

Abkürzende Bezeichnung für die grundlegenden Eigenschaften von *Transaktionen*: Atomicity, Consistency, Isolation und Durability.

Administration
administration

Verwaltung und Steuerung einer *UTM-Anwendung* durch einen *Administrator* oder ein *Administrationsprogramm*.

Administrations-Journal
administration journal

siehe *Cluster-Administrations-Journal*.

Administrationskommando
administration command

Kommandos, mit denen der *Administrator* einer *UTM-Anwendung* Administrationsfunktionen für diese Anwendung durchführt. Die Administrationskommandos sind als *Transaktionscodes* realisiert.

Administrationsprogramm
administration program

Teilprogramm, das Aufrufe der *Programmschnittstelle für die Administration* enthält. Dies kann das Standard-Administrationsprogramm *KDCADM* sein, das mit openUTM ausgeliefert wird, oder ein vom Anwender selbst erstelltes Programm.

Administrator
administrator

Benutzer mit Administrationsberechtigung.

AES

AES (Advanced Encryption Standard) ist der aktuelle symmetrische Verschlüsselungsstandard, festgelegt vom NIST (National Institute of Standards and Technology), basierend auf dem an der Universität Leuven (B) entwickelten Rijndael-Algorithmus. Wird das AES-Verfahren verwendet, dann erzeugt der UPIC-Client für jede Sitzung einen AES-Schlüssel.

Akzeptor (CPI-C)
acceptor

Die Kommunikationspartner einer *Conversation* werden *Initiator* und Akzeptor genannt. Der Akzeptor nimmt die vom Initiator eingeleitete Conversation mit Accept_Conversation entgegen.

Anmelde-Vorgang (KDCS)
sign-on service

Spezieller *Dialog-Vorgang*, bei dem die Anmeldung eines Benutzers an eine UTM-Anwendung durch *Teilprogramme* gesteuert wird.

Anschlussprogramm
linkage program

siehe *KDCROOT*.

Anwendungsinformation
application information

Sie stellt die Gesamtmenge der von der *UTM-Anwendung* benutzten Daten dar. Dabei handelt es sich um Speicherbereiche und Nachrichten der UTM-Anwendung, einschließlich der aktuell auf dem Bildschirm angezeigten Daten. Arbeitet die UTM-Anwendung koordiniert mit einem Datenbanksystem, so gehören die in der Datenbank gespeicherten Daten ebenfalls zur Anwendungsinformation.

Anwendungs-Kaltstart
application cold start

siehe *Kaltstart*.

Anwendungsprogramm
application program

Ein Anwendungsprogramm bildet den Hauptbestandteil einer *UTM-Anwendung*. Es besteht aus der Main Routine *KDCROOT* und den *Teilprogrammen*. Es bearbeitet alle Aufträge, die an eine *UTM-Anwendung* gerichtet werden.

Anwendungs-Warmstart
application warm start

siehe *Warmstart*.

Apache Axis

Apache Axis (Apache eXtensible Interaction System) ist eine SOAP-Engine zur Konstruktion von darauf basierenden Web Services und Client-Anwendungen. Es existiert eine Implementierung in C++ und Java.

Apache Tomcat

Apache Tomcat stellt eine Umgebung zur Ausführung von Java-Code auf Web-Servern bereit, die im Rahmen des Jakarta-Projekts der Apache Software Foundation entwickelt wird. Es handelt sich um einen in Java geschriebenen Servlet-Container, der mithilfe des JSP-Compilers Jasper auch JavaServer Pages in Servlets übersetzen und ausführen kann. Dazu kommt ein kompletter HTTP-Server.

Application Context (OSI)
application context

Der Application Context ist die Menge der Regeln, die für die Kommunikation zwischen zwei Anwendungen gelten sollen. Dazu gehören z.B. die *abstrakten Syntaxen* und die zugeordneten *Transfer-Syntaxen*.

Application Entity (OSI)
application entity

Eine Application Entity (AE) repräsentiert alle für die Kommunikation relevanten Aspekte einer realen Anwendung. Eine Application Entity wird durch einen global (d.h. weltweit) eindeutigen Namen identifiziert, den *Application Entity Title* (AET). Jede Application Entity repräsentiert genau einen *Application Process*. Ein Application Process kann mehrere Application Entities umfassen.

Application Entity Qualifier (OSI)
application entity qualifier

Bestandteil des *Application Entity Titles*. Der Application Entity Qualifier identifiziert einen *Dienstzugriffspunkt* innerhalb der Anwendung. Ein Application Entity Qualifier kann unterschiedlich aufgebaut sein. openUTM unterstützt den Typ "Zahl".

Application Entity Title (OSI)
application entity title

Ein Application Entity Title ist ein global (d.h. weltweit) eindeutiger Name für eine *Application Entity*. Er setzt sich zusammen aus dem *Application Process Title* des jeweiligen *Application Process* und dem *Application Entity Qualifier.*

Application Process (OSI)
application process

Der Application Process repräsentiert im *OSI-Referenzmodell* eine Anwendung. Er wird durch den *Application Process Title* global (d.h. weltweit) eindeutig identifiziert.

Application Process Title (OSI)
application process title

Gemäß der OSI-Norm dient der Application Process Title (APT) zur global (d.h. weltweit) eindeutigen Identifizierung von Anwendungen. Er kann unterschiedlich aufgebaut sein. openUTM unterstützt den Typ *Object Identifier*.

Application Service Element (OSI)
application service element

Ein Application Service Element (ASE) repräsentiert eine Funktionsgruppe der Anwendungsschicht (Schicht 7) des *OSI-Referenzmodells*.

Association (OSI)
association

Eine Association ist eine Kommunikationsbeziehung zwischen zwei *Application Entities*. Dem Begriff Association entspricht der *LU6.1*-Begriff *Session*.

Asynchron-Auftrag
queued job

Auftrag, der vom Auftraggeber zeitlich entkoppelt durchgeführt wird. Zur Bearbeitung von Asynchron-Aufträgen sind in openUTM *Message Queuing* Funktionen integriert, vgl. *UTM-gesteuerte Queue* und *Service-gesteuerte Queue*. Ein Asynchron-Auftrag wird durch die *Asynchron-Nachricht*, den Empfänger und ggf. den gewünschten Ausführungszeitpunkt beschrieben.
Ist der Empfänger ein Terminal, ein Drucker oder eine Transportsystem-Anwendung, so ist der Asynchron-Auftrag ein *Ausgabe-Auftrag*; ist der Empfänger ein Asynchron-Vorgang derselben oder einer fernen Anwendung, so handelt es sich um einen *Hintergrund-Auftrag*.
Asynchron-Aufträge können *zeitgesteuerte Aufträge* sein oder auch in einen *Auftrags-Komplex* integriert sein.

Asynchron-Conversation
asynchronous conversation

CPI-C-Conversation, bei der nur der *Initiator* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein asynchroner Transaktionscode generiert sein.

Asynchron-Nachricht
asynchronous message

Asynchron-Nachrichten sind Nachrichten, die an eine *Message Queue* gerichtet sind. Sie werden von der lokalen *UTM-Anwendung* zunächst zwischengespeichert und dann unabhängig vom Auftraggeber weiter verarbeitet. Je nach Empfänger unterscheidet man folgende Typen von Asynchron-Nachrichten:

- Bei Asynchron-Nachrichten an eine *UTM-gesteuerte Queue* wird die Weiterverarbeitung komplett durch openUTM gesteuert. Zu diesem Typ gehören Nachrichten, die einen lokalen oder fernen *Asynchron-Vorgang* starten (vgl. auch *Hintergrund-Auftrag*) und Nachrichten, die zur Ausgabe an ein Terminal, einen Drucker oder eine Transportsystem-Anwendung geschickt werden (vgl. auch *Ausgabe-Auftrag*).
- Bei Asynchron-Nachrichten an eine *Service-gesteuerte Queue* wird die Weiterverarbeitung durch einen *Service* der Anwendung gesteuert. Zu diesem Typ gehören Nachrichten an eine *TAC-Queue*, Nachrichten an eine *USER-Queue* und Nachrichten an eine *Temporäre Queue*. Die User-Queue und die Temporäre Queue müssen dabei zur lokalen Anwendung gehören, die TAC-Queue kann sowohl in der lokalen als auch in einer fernen Anwendung liegen.

Asynchron-Programm
asynchronous program

Teilprogramm, das von einem *Hintergrund-Auftrag* gestartet wird.

Asynchron-Vorgang (KDCS)
asynchronous service

Vorgang , der einen *Hintergrund-Auftrag* bearbeitet. Die Verarbeitung erfolgt entkoppelt vom Auftraggeber. Ein Asynchron-Vorgang kann aus einem oder mehreren Teilprogrammen /Transaktionen bestehen. Er wird über einen asynchronen *Transaktionscode* gestartet.

Auftrag
job

Anforderung eines *Services* , der von einer *UTM-Anwendung* zur Verfügung gestellt wird, durch Angabe eines *Transaktionscodes* . Siehe auch: *Ausgabe-Auftrag* , *Dialog-Auftrag* , *Hintergrund-Auftrag* , *Auftrags-Komplex*.

Auftraggeber-Vorgang
job-submitting service

Ein Auftraggeber-Vorgang ist ein *Vorgang*, der zur Bearbeitung eines Auftrags einen Service von einer anderen Server-Anwendung (*Auftragnehmer-Vorgang*) anfordert.

Auftragnehmer-Vorgang
job-receiving service

Ein Auftragnehmer-Vorgang ist ein *Vorgang*, der von einem *Auftraggeber-Vorgang* einer anderen Server-Anwendung gestartet wird.

Auftrags-Komplex
job complex

Auftrags-Komplexe dienen dazu, *Asynchron-Aufträgen Quittungsaufträge* zuzuordnen. Ein Asynchron-Auftrag innerhalb eines Auftrags-Komplexes wird *Basis-Auftrag* genannt.

Ausgabe-Auftrag
queued output job

Ausgabeaufträge sind *Asynchron-Aufträge*, die die Aufgabe haben, eine Nachricht, z.B. ein Dokument, an einen Drucker, ein Terminal oder eine Transportsystem-Anwendung auszugeben. Ausgabeaufträge werden ausschließlich von UTM-Systemfunktionen bearbeitet, d.h. für die Bearbeitung müssen keine Teilprogramme erstellt werden.

Authentisierung
authentication

siehe *Zugangskontrolle*.

Autorisierung
authorization

siehe *Zugriffskontrolle*.

Axis

siehe *Apache Axis*.

Basis-Auftrag
basic job

Asynchron-Auftrag in einem *Auftrags-Komplex*.

Basisformat
basic format

Format, in das der Terminal-Benutzer alle Angaben eintragen kann, die notwendig sind, um einen Vorgang zu starten.

Basisname
filebase

Basisname der UTM-Anwendung.
Auf BS2000-Systemen ist Basisname das Präfix für die *KDCFILE*, die *Benutzerprotokoll-Datei* USLOG und die *System-Protokolldatei* SYSLOG.
Auf Unix-, Linux- und Windows-Systemen ist Basisname der Name des Verzeichnisses, unter dem die KDCFILE, die Benutzerprotokoll-Datei USLOG, die System-Protokolldatei SYSLOG und weitere Dateien der UTM-Anwendung abgelegt sind.

Basisname der Knoten-Anwendung
node filebase

Dateinamens-Präfix bzw. Verzeichnisname für die *KDCFILE*, *Benutzerprotokoll-Datei* und *Systemprotokoll-Datei* der *Knoten-Anwendung*.

Basisname der UTM-Cluster-Anwendung
cluster filebase

Dateinamens-Präfix bzw. Verzeichnisname für die *UTM-Cluster-Dateien*.

Benutzerausgang
user exit

Begriff ersetzt durch *Event-Exit.*

Benutzerkennung
user ID

Bezeichner für einen Benutzer, der in der *Konfiguration* der *UTM-Anwendung* festgelegt ist (optional mit Passwort zur *Zugangskontrolle*) und dem spezielle Zugriffsrechte (*Zugriffskontrolle*) zugeordnet sind. Ein Terminal-Benutzer muss bei der Anmeldung an die UTM-Anwendung diesen Bezeichner (und ggf. das zugeordnete Passwort) angeben. Auf BS2000-Systemen ist außerdem eine Zugangskontrolle über *Kerberos* möglich.
Für andere Clients ist die Angabe der Benutzerkennung optional, siehe auch *Verbindungs-Benutzerkennung*.
UTM-Anwendungen können auch ohne Benutzerkennungen generiert werden.

Benutzer-Protokolldatei
user log file

Datei oder Dateigeneration, in die der Benutzer mit dem KDCS-Aufruf LPUT Sätze variabler Länge schreibt. Jedem Satz werden die Daten aus dem KB-Kopf des *KDCS-Kommunikationsbereichs* vorangestellt. Die Benutzerprotokolldatei unterliegt der Transaktionssicherung von openUTM.

Berechtigungsprüfung
sign-on check

siehe *Zugangskontrolle*.

Beweissicherung (BS2000-Systeme)
audit

Im Betrieb einer *UTM-Anwendung* können zur Beweissicherung sicherheitsrelevante UTM-Ereignisse von *SAT* protokolliert werden.

Bildschirm-Wiederanlauf
screen restart

Wird ein *Dialog-Vorgang* unterbrochen, gibt openUTM beim *Vorgangswiederanlauf* die *Dialog-Nachricht* der letzten abgeschlossenen *Transaktion* erneut auf dem Bildschirm aus, sofern die letzte Transaktion eine Nachricht auf den Bildschirm ausgegeben hat.

Browsen von Asynchron-Nachrichten
browsing asynchronous messages

Ein *Vorgang* liest nacheinander die *Asynchron-Nachrichten*, die sich in einer *Service-gesteuerten Queue* befinden. Die Nachrichten werden während des Lesens nicht gesperrt und verbleiben nach dem Lesen in der Queue. Dadurch ist gleichzeitiges Lesen durch unterschiedliche Vorgänge möglich.

Bypass-Betrieb (BS2000-Systeme)
bypass mode

Betriebsart eines Druckers, der lokal an ein Terminal angeschlossen ist. Im Bypass-Betrieb wird eine an den Drucker gerichtete *Asynchron-Nachricht* an das Terminal gesendet und von diesem auf den Drucker umgeleitet, ohne auf dem Bildschirm angezeigt zu werden.

Cache-Speicher
cache

Pufferbereich zur Zwischenspeicherung von Anwenderdaten für alle Prozesse einer *UTM-Anwendung*. Der Cache-Speicher dient zur Optimierung der Zugriffe auf den *Pagepool* und für UTM-Cluster-Anwendungen zusätzlich auf den *Cluster-Pagepool*.

CCR (Commitment, Concurrency and Recovery)

CCR ist ein von OSI definiertes Application Service Element (ASE) für die OSI-TP-Kommunkation, welches die Protokollelemente (Services) zum Beginn und Abschluss (Commit oder Rollback) einer *Transaktion* enthält. CCR unterstützt das Zwei-Phasen-Commitment.

CCS-Name (BS2000-Systeme)
CCS name

siehe *Coded-Character-Set-Name* .

Client
client

Clients einer *UTM-Anwendung* können sein:

- Terminals
- UPIC-Client-Programme
- Transportsystem-Anwendungen (z.B. DCAM-, PDN-, CMX-, Socket-Anwendungen oder UTM-Anwendungen, die als *Transportsystem-Anwendung* generiert sind)

Clients werden über LTERM-Partner an die UTM-Anwendung angeschlossen.
Hinweis: UTM-Clients mit Trägersystem OpenCPIC werden wie *OSI TP-Partner* behandelt.

Client-Seite einer Conversation
client side of a conversation

Begriff ersetzt durch *Initiator*.

Cluster

Eine Anzahl von Rechnern, die über ein schnelles Netzwerk verbunden sind und die von außen in vielen Fällen als ein Rechner gesehen werden können. Das Ziel des "Clustering" ist meist die Erhöhung der Rechenkapazität oder der Verfügbarkeit gegenüber einem einzelnen Rechner.

Cluster-Administrations-Journal
cluster administration journal

Das Cluster-Administrations-Journal besteht aus:

- zwei Protokolldateien mit Endungen JRN1 und JRN2 für globale Administrationsaktionen,
- der JKAA-Datei, die eine Kopie der KDCS Application Area (KAA) enthält. Aus dieser Kopie werden administrative Änderungen übernommen, die nicht mehr in den beiden Protokolldateien enthalten sind.

Die Administrations-Journal-Dateien dienen dazu, administrative Aktionen, die in einer UTM-Cluster-Anwendung Cluster-weit auf alle Knoten-Anwendungen wirken sollen, an die anderen Knoten-Anwendungen weiterzugeben.

Cluster-GSSB-Datei
cluster GSSB file

Datei zur Verwaltung von GSSBs in einer *UTM-Cluster-Anwendung*. Die Cluster-GSSB-Datei wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Cluster-Konfigurationsdatei
cluster configuration file

Datei, die die zentralen Konfigurationsdaten einer *UTM-Cluster-Anwendung* enthält. Die Cluster-Konfigurationsdatei wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Cluster-Lock-Datei
cluster lock file

Datei einer *UTM-Cluster-Anwendung*, die dazu dient, Knoten-übergreifende Sperren auf Anwenderdatenbereiche zu verwalten.

Cluster-Pagepool
cluster pagepool

Der Cluster-Pagepool besteht aus einer Verwaltungsdatei und bis zu 10 Dateien, in denen die Cluster-weit verfügbaren Anwenderdaten (Vorgangsdaten inklusive LSSB, GSSB und ULS) einer *UTM-Cluster-Anwendung* gespeichert werden. Der Cluster-Pagepool wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Cluster-Startserialisierungs-Datei
cluster start serialization file

Lock-Datei, mit der die Starts einzelner Knoten-Anwendungen serialisiert werden (nur auf Unix-, Linux- und Windows-Systemen).

Cluster-ULS-Datei
cluster ULS file

Datei zur Verwaltung von ULS-Bereichen einer *UTM-Cluster-Anwendung*. Die Cluster-ULS-Datei wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Cluster-User-Datei
cluster user file

Datei, die die Verwaltungsdaten der Benutzer einer *UTM-Cluster-Anwendung* enthält. Die Cluster-User-Datei wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Coded-Character-Set-Name (BS2000-Systeme)
coded character set name

Bei Verwendung des Produkts *XHCS* (e**X**tended **H**ost **C**ode **S**upport) wird jeder verwendete Zeichensatz durch einen Coded-Character-Set-Namen (abgekürzt: "CCS-Name" oder "CCSN") eindeutig identifiziert.

Communication Resource Manager
communication resource manager

Communication Resource Manager (CRMs) kontrollieren in verteilten Systemen die Kommunikation zwischen den Anwendungsprogrammen. openUTM stellt CRMs für den internationalen Standard OSI TP, für den Industrie-Standard *LU6.1* und für das openUTM-eigene Protokoll UPIC zur Verfügung.

Contention Loser
contention loser

Jede Verbindung zwischen zwei Partnern wird von einem der Partner verwaltet. Der Partner, der die Verbindung verwaltet, heißt *Contention Winner*. Der andere Partner ist der Contention Loser.

Contention Winner
contention winner

Der Contention Winner einer Verbindung übernimmt die Verwaltung der Verbindung. Aufträge können sowohl vom Contention Winner als auch vom *Contention Loser* gestartet werden. Im Konfliktfall, wenn beide Kommunikationspartner gleichzeitig einen Auftrag starten wollen, wird die Verbindung vom Auftrag des Contention Winner belegt.

Conversation
conversation

Bei CPI-C nennt man die Kommunikation zwischen zwei CPI-C-Anwendungsprogrammen Conversation. Die Kommunikationspartner einer Conversation werden *Initiator* und *Akzeptor* genannt.

Conversation-ID
conversation ID

Jeder *Conversation* wird von CPI-C lokal eine Conversation-ID zugeordnet, d.h. *Initiator* und *Akzeptor* haben jeweils eine eigene Conversation-ID. Mit der Conversation-ID wird jeder CPI-C-Aufruf innerhalb eines Programms eindeutig einer Conversation zugeordnet.

CPI-C

CPI-C (**C**ommon **P**rogramming **I**nterface for **C**ommunication) ist eine von X/Open und dem CIW (**C**PI-C **I**mplementor's **W**orkshop) normierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen. Das in openUTM implementierte CPI-C genügt der CPI-C V2.0 CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung.
CPI-C in openUTM kann über die Protokolle OSI TP, LU6.1, UPIC und mit openUTM-LU6.2 kommunizieren.

Cross Coupled System / XCS

Verbund von BS2000-Rechnern mit *Highly Integrated System Complex* Multiple System Control Facility (HIPLEX® MSCF).

Datenraum (BS2000-Systeme)
data space

Virtueller Adressraum des BS2000, der in seiner gesamten Größe vom Anwender genutzt werden kann.
In einem Datenraum können nur Daten und als Daten abgelegte Programme adressiert werden, es kann kein Programmcode zum Ablauf gebracht werden.

Dead Letter Queue
dead letter queue

Die Dead Letter Queue ist eine *TAC-Queue* mit dem festen Namen KDCDLETQ. Sie steht immer zur Verfügung, um Asynchron-Nachrichten an *Transaktionscodes*, TAC-Queues, LPAP- oder OSI-LPAP-Partner zu sichern, die nicht verarbeitet werden konnten.
Die Sicherung von Asynchron-Nachrichten in der Dead Letter Queue kann durch den Parameter DEAD-LETTER-Q der TAC-, LPAP- oder OSI-LPAP-Anweisung für jedes Nachrichtenziel einzeln ein- und ausgeschaltet werden.

DES

DES (Data Encryption Standard) ist eine internationale Norm zur Verschlüsselung von Daten. Bei diesem Verfahren wird ein Schlüssel zum Ver- und Entschlüsseln verwendet. Wird das DES-Verfahren verwendet, dann erzeugt der UPIC-Client für jede Sitzung einen DES-Schlüssel.

Dialog-Auftrag
dialog job, interactive job

Auftrag, der einen *Dialog-Vorgang* startet. Der Auftrag kann von einem *Client* oder - bei *Server-Server-Kommunikation* - von einer anderen Anwendung erteilt werden.

Dialog-Conversation
dialog conversation

CPI-C-Conversation, bei der sowohl der *Initiator* als auch der *Akzeptor* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein Dialog-Transaktionscode generiert sein.

Dialog-Nachricht
dialog message

Nachricht, die eine Antwort erfordert oder selbst eine Antwort auf eine Anfrage ist. Dabei bilden Anfrage und Antwort einen *Dialog-Schritt*.

Dialog-Programm
dialog program

Teilprogramm, das einen *Dialog-Schritt* teilweise oder vollständig bearbeitet.

Dialog-Schritt
dialog step

Ein Dialog-Schritt beginnt mit dem Empfang einer *Dialog-Nachricht* durch die *UTM-Anwendung*. Er endet mit der Antwort der UTM-Anwendung.

Dialog-Terminalprozess (Unix-, Linux- und Windows-Systeme)
dialog terminal process

Ein Dialog-Terminalprozess verbindet ein Unix-, Linux- oder Windows-Terminal mit den *Workprozessen* der *UTM-Anwendung*. Dialog-Terminalprozesse werden entweder vom Benutzer durch Eingabe von utmdtp oder über die LOGIN-Shell gestartet. Für jedes Terminal, das an eine UTM-Anwendung angeschlossen werden soll, ist ein eigener Dialog-Terminalprozess erforderlich.

Dialog-Vorgang
dialog service

Vorgang , der einen *Auftrag* im Dialog (zeitlich gekoppelt) mit dem Auftraggeber (*Client* oder eine andere Server-Anwendung) bearbeitet. Ein Dialog-Vorgang verarbeitet *Dialog-Nachrichten* vom Auftraggeber und erzeugt Dialog-Nachrichten für diesen. Ein Dialog-Vorgang besteht aus mindestens einer *Transaktion.* Ein Dialog-Vorgang umfasst in der Regel mindestens einen *Dialog-Schritt*. Ausnahme: Bei *Vorgangskettung* können auch mehrere Vorgänge einen Dialog-Schritt bilden.

Dienst
service

Programm auf Windows-Systemen, das im Hintergrund unabhängig von angemeldeten Benutzern oder Fenstern abläuft.

Dienstzugriffspunkt
service access point

Im *OSI-Referenzmodell* stehen einer Schicht am Dienstzugriffspunkt die Leistungen der darunterliegenden Schicht zur Verfügung. Der Dienstzugriffspunkt wird im lokalen System durch einen *Selektor* identifiziert. Bei der Kommunikation bindet sich die *UTM-Anwendung* an einen Dienstzugriffspunkt. Eine Verbindung wird zwischen zwei Dienstzugriffspunkten aufgebaut.

Distributed Transaction Processing

X/Open-Architekturmodell für die transaktionsorientierte *verteilte Verarbeitung*.

Druckadministration
print administration

Funktionen zur *Drucksteuerung* und Administration von *Ausgabeaufträgen*, die an einen Drucker gerichtet sind.

Druckerbündel
printer pool

Mehrere Drucker, die demselben *LTERM-Partner* zugeordnet sind.

Druckergruppe (Unix- und Linux-Systeme)
printer group

Die Unix- oder Linux-Plattform richtet für jeden Drucker standardmäßig eine Druckergruppe ein, die genau diesen Drucker enthält. Darüber hinaus lassen sich mehrere Drucker einer Druckergruppe, aber auch ein Drucker mehreren Druckergruppen zuordnen.

Druckerprozess (Unix- und Linux-Systeme)
printer process

Prozess, der vom *Mainprozess* zur Ausgabe von *Asynchron-Nachrichten* an eine *Druckergruppe* eingerichtet wird. Er existiert, solange die Druckergruppe an die *UTM-Anwendung* angeschlossen ist. Pro angeschlossener Druckergruppe gibt es einen Druckerprozess.

Druckersteuerstation
printer control terminal

Begriff wurde ersetzt durch *Druckersteuer-LTERM*.

Druckersteuer-LTERM
printer control LTERM

Über ein Druckersteuer-LTERM kann sich ein *Client* oder ein Terminal-Benutzer an eine *UTM-Anwendung* anschließen. Von dem Client-Programm oder Terminal aus kann dann die *Administration* der Drucker erfolgen, die dem Druckersteuer-LTERM zugeordnet sind. Hierfür ist keine Administrationsberechtigung notwendig.

Drucksteuerung
print control

openUTM-Funktionen zur Steuerung von Druckausgaben.

Dynamische Konfiguration
dynamic configuration

Änderung der *Konfiguration* durch die Administration. Im laufenden Betrieb der Anwendung können UTM-Objekte wie z.B. *Teilprogramme*, *Transaktionscodes*, *Clients*, *LU6.1-Verbindungen*, Drucker oder *Benutzerkennungen* in die Konfiguration aufgenommen, modifiziert oder teilweise auch gelöscht werden. Hierzu können die Administrationsprogramme WinAdmin oder WebAdmin verwendet werden, oder es müssen eigene *Administrationsprogramme* erstellt werden, die die Funktionen der *Programmschnittstelle der Administration* nutzen.

Einschritt-Transaktion
single-step transaction

Transaktion, die genau einen *Dialog-Schritt* umfasst.

Einschritt-Vorgang
single-step service

Dialog-Vorgang, der genau einen *Dialog-Schritt* umfasst.

Ereignisgesteuerter Vorgang
event-driven service

Begriff ersetzt durch *Event-Service*.

Event-Exit
event exit

Routine des *Anwendungsprogramms*, das bei bestimmten Ereignissen (z.B. Start eines Prozesses, Ende eines Vorgangs) automatisch gestartet wird. Diese darf - im Gegensatz zu den *Event-Services* - keine KDCS-, CPI-C- und XATMI-Aufrufe enthalten.

Event-Funktion
event function

Oberbegriff für *Event-Exits* und *Event-Services*.

Event-Service
event service

Vorgang, der beim Auftreten bestimmter Ereignisse gestartet wird, z.B. bei bestimmten UTM-Meldungen. Die *Teilprogramme* ereignisgesteuerter Vorgänge müssen KDCS-Aufrufe enthalten.

Funktionseinheit, Functional Unit (FU)
functional unit

Teilmenge des *OSI-TP*-Protokolls, die eine bestimmte Funktionalität beinhaltet. Das OSI-TP-Protokoll ist in folgende Funktionseinheiten aufgeteilt:

- Dialogue
- Shared Control
- Polarized Control
- Handshake
- Commit
- Chained Transactions
- Unchained Transactions
- Recovery

Ein Hersteller, der OSI-TP implementiert, muss nicht alle Funktionseinheiten realisieren, sondern kann sich auf eine Teilmenge beschränken. Eine Kommunikation zwischen Anwendungen zweier unterschiedlicher OSI-TP-Implementierungen ist nur dann möglich, wenn die realisierten Funktionseinheiten zueinander passen.

Generierung
generation

siehe *UTM-Generierung*.

Globaler Sekundärer Speicherbereich/GSSB
global secondary storage area

siehe *Sekundärspeicherbereich.*

Hardcopy-Betrieb
hardcopy mode

Betriebsart eines Druckers, der lokal an ein Terminal angeschlossen ist. Dabei wird eine Nachricht, die auf dem Bildschirm angezeigt wird, zusätzlich auf dem Drucker abgedruckt.

Heterogene Kopplung
heterogeneous link

Bei *Server-Server-Kommunikation:* Kopplung einer *UTM-Anwendung* mit einer Nicht-UTM-Anwendung, z.B. einer CICS- oder TUXEDO-Anwendung.

Highly Integrated System Complex / HIPLEX ®

Produktfamilie zur Realisierung eines Bedien-, Last- und Verfügbarkeitsverbunds mit mehreren BS2000-Servern.

Hintergrund-Auftrag
background job

Hintergrund-Aufträge sind *Asynchron-Aufträge*, die an einen *Asynchron-Vorgang* der eigenen oder einer fernen Anwendung gerichtet sind. Hintergrund-Aufträge eignen sich besonders für zeitintensive oder zeitunkritische Verarbeitungen, deren Ergebnis keinen direkten Einfluss auf den aktuellen Dialog hat.

HIPLEX ® MSCF

(MSCF = **M**ultiple **S**ystem **C**ontrol **F**acility)
stellt bei HIPLEX® die Infrastruktur sowie Basisfunktionen für verteilte Anwendungen bereit.

Homogene Kopplung
homogeneous link

Bei *Server-Server-Kommunikation*: Kopplung von *UTM-Anwendungen.* Dabei spielt es keine Rolle, ob die Anwendungen auf der gleichen oder auf unterschiedlichen Betriebssystem-Plattformen ablaufen.

Inbound-Conversation (CPI-C)
inbound conversation

siehe *Incoming-Conversation.*

Incoming-Conversation (CPI-C)
incoming conversation

Eine *Conversation*, bei der das lokale CPI-C-Programm *Akzeptor* ist, heißt Incoming-Conversation. In der X/Open-Specification wird für Incoming-Conversation auch das Synonym Inbound-Conversation verwendet.

Initiale KDCFILE
initial KDCFILE

In einer *UTM-Cluster-Anwendung* die *KDCFILE*, die von *KDCDEF* erzeugt wurde und vor dem Start der Knoten-Anwendungen für jeden Knoten kopiert werden muss.

Initiator (CPI-C)
initiator

Die Kommunikationspartner einer *Conversation* werden Initiator und *Akzeptor* genannt. Der Initiator baut die Conversation mit den CPI-C-Aufrufen Initialize_Conversation und Allocate auf.

Insert
insert

Feld in einem Meldungstext, in das openUTM aktuelle Werte einträgt.

Inverser KDCDEF
inverse KDCDEF

Funktion, die aus den Konfigurationsdaten der *KDCFILE*, die im laufenden Betrieb dynamisch angepasst wurde, Steueranweisungen für einen *KDCDEF*-Lauf erzeugt. Der inverse KDCDEF kann "offline" unter KDCDEF oder "online" über die *Programmschnittstelle zur Administration* gestartet werden.

IUTMDB
IUTMDB

Schnittstelle für die koordinierte Zusammenarbeit mit externen Resource Managern auf BS2000-Systemen. Dazu gehören Datenhaltungssysteme (LEASY) und Datenbanksysteme (SESAM/SQL, UDS/SQL).

JConnect-Client
JConnect client

Bezeichung für Clients auf Basis des Produkts openUTM-JConnect. Die Kommunikation mit der UTM-Anwendung erfolgt über das *UPIC-Protokoll*.

JDK

Java Development Kit
Standard-Entwicklungsumgebung von Oracle Corporation für die Entwicklung von Java-Anwendungen.

Kaltstart
cold start

Starten einer *UTM-Anwendung* nach einer *normalen Beendigung* der Anwendung oder nach einer Neugenerierung (vgl. auch *Warmstart*).

KDCADM

Standard-Administrationsprogramm, das zusammen mit openUTM ausgeliefert wird. KDCADM stellt Administrationsfunktionen zur Verfügung, die über Transaktionscodes (*Administrationskommandos*) aufgerufen werden.

KDCDEF

UTM-Tool für die *Generierung* von *UTM-Anwendungen*. KDCDEF erstellt anhand der Konfigurationsinformationen in den KDCDEF-Steueranweisungen die UTM-Objekte *KDCFILE* und die ROOT-Tabellen-Source für die Main Routine *KDCROOT*.
In UTM-Cluster-Anwendungen erstellt KDCDEF zusätzlich die *Cluster-Konfigurationsdatei,* die *Cluster-User-Datei*, den *Cluster-Pagepool*, die *Cluster-GSSB-Datei* und die *Cluster-ULS-Datei*.

KDCFILE

Eine oder mehrere Dateien, die für den Ablauf einer *UTM-Anwendung* notwendige Daten enthalten. Die KDCFILE wird mit dem UTM-Generierungstool *KDCDEF* erstellt. Die KDCFILE enthält unter anderem die *Konfiguration* der Anwendung.

KDCROOT

Main Routine eines *Anwendungsprogramms*, die das Bindeglied zwischen *Teilprogrammen* und UTM-Systemcode bildet. KDCROOT wird zusammen mit den *Teilprogrammen* zum *Anwendungsprogramm* gebunden.

KDCS-Parameterbereich
KDCS parameter area

siehe *Parameterbereich.*

KDCS-Programmschnittstelle
KDCS program interface

Universelle UTM-Programmschnittstelle, die den nationalen Standard DIN 66 265 erfüllt und Erweiterungen enthält. Mit KDCS (Kompatible Datenkommunikationsschnittstelle) lassen sich z.B. Dialog-Services erstellen und *Message Queuing* Funktionen nutzen. Außerdem stellt KDCS Aufrufe zur *verteilten Verarbeitung* zur Verfügung.

Kerberos

Kerberos ist ein standardisiertes Netzwerk-Authentisierungsprotokoll (RFC1510), das auf kryptographischen Verschlüsselungsverfahren basiert, wobei keine Passwörter im Klartext über das Netzwerk gesendet werden.

Kerberos-Principal
Kerberos principal

Eigentümer eines Schlüssels.
Kerberos arbeitet mit symmetrischer Verschlüsselung, d.h. alle Schlüssel liegen an zwei Stellen vor, beim Eigentümer eines Schlüssels (Principal) und beim KDC (Key Distribution Center).

Keycode
key code

Code, der in einer Anwendung eine bestimmte Zugriffsberechtigung oder eine bestimmte Rolle repräsentiert. Mehrere Keycodes werden zu einem *Keyset* zusammengefasst.

Keyset
key set

Zusammenfassung von einem oder mehrerer *Keycodes* unter einem bestimmten Namen. Ein Keyset definiert Berechtigungen im Rahmen des verwendeten Berechtigungskonzepts (Lock-/Keycode-Konzept oder *Access-List* -Konzept).
Ein Keyset kann einer *Benutzerkennung* , einem *LTERM-Partner* , einem *(OSI-) LPAP-Partner* , einem *Service* oder einer *TAC-Queue* zugeordnet werden.

Knoten
node

Einzelner Rechner eines *Clusters*.

Knoten-Anwendung
node application

UTM-Anwendung, die als Teil einer *UTM-Cluster-Anwendung* auf einem einzelnen *Knoten* zum Ablauf kommt.

Knoten-Recovery
node recovery

Wenn für eine abnormal beendete Knoten-Anwendung zeitnah kein Warmstart auf ihrem eigenen *Knoten-Rechner* möglich ist, kann man für diesen Knoten auf einem anderen Knoten des UTM-Clusters eine Knoten-Recovery (Wiederherstellung) durchführen. Dadurch können Sperren, die von der ausgefallenen Knoten-Anwendung gehalten werden, freigegeben werden, um die laufende *UTM-Cluster-Anwendung* nicht unnötig zu beeinträchtigen.

Knotengebundener Vorgang
node bound service

Ein knotengebundener Vorgang eines Benutzers kann nur an der Knoten-Anwendung fortgesetzt werden, an der der Benutzer zuletzt angemeldet war. Folgende Vorgänge sind immer knotengebunden:

- Vorgänge, die eine Kommunikation mit einem Auftragnehmer über LU6.1 oder OSI TP begonnen haben und bei denen der Auftragnehmervorgang noch nicht beendet wurde
- eingeschobene Vorgänge einer Vorgangskellerung
- Vorgänge, die eine SESAM-Transaktion abgeschlossen haben

Außerdem ist der Vorgang eines Benutzers knotengebunden, solange der Benutzer an einer Knoten-Anwendung angemeldet ist.

Kommunikationsbereich/KB (KDCS)
communication area

Transaktionsgesicherter KDCS-*Primärspeicherbereich*, der Vorgangs-spezifische Daten enthält. Der Kommunikationsbereich besteht aus 3 Teilen:

- dem KB-Kopf mit allgemeinen Vorgangsdaten
- dem KB-Rückgabebereich für Rückgaben nach KDCS-Aufrufen
- dem KB-Programmbereich zur Datenübergabe zwischen UTM-Teilprogrammen innerhalb eines *Vorgangs*.

Kommunikationsendpunkt
communication end point

siehe *Transportsystem-Endpunkt*

Konfiguration
configuration

Summe aller Eigenschaften einer *UTM-Anwendung*. Die Konfiguration beschreibt:

- Anwendungs- und Betriebsparameter
- die Objekte der Anwendung und die Eigenschaften dieser Objekte. Objekte sind z.B. *Teilprogramme* und *Transaktionscodes*, Kommunikationspartner, Drucker, *Benutzerkennungen*
- definierte Zugriffsschutz- und Zugangsschutzmaßnahmen

Die Konfiguration einer UTM-Anwendung wird bei der UTM-Generierung festgelegt (*statische Konfiguration*) und kann per *Administration* dynamisch (während des Anwendungslaufs) geändert werden (*dynamische Konfiguration*). Die Konfiguration ist in der *KDCFILE* abgelegt.

Logging-Prozess
logging process

Prozess auf Unix-, Linux- und Windows-Systemen, der die Protokollierung von Abrechnungssätzen oder Messdaten steuert.

Logische Verbindung
virtual connection

Zuordnung zweier Kommunikationspartner.

Log4j

Log4j ist ein Teil des Apache Jakarta Projekts. Log4j bietet Schnittstellen zum Protokollieren von Informationen (Ablauf-Informationen, Trace-Records,...) und zum Konfigurieren der Protokoll-Ausgabe. *WS4UTM* verwendet das Softwareprodukt Log4j für die Trace- und Logging-Funktionalität.

Lockcode

Code, um einen LTERM-Partner oder einen Transaktionscode vor unberechtigtem Zugriff zu schützen. Damit ist ein Zugriff nur möglich, wenn das *Keyset* des Zugreifenden den passenden *Keycode* enthält (Lock-/Keycode-Konzept).

Lokaler Sekundärer Speicherbereich/LSSB
local secondary storage area

siehe *Sekundärspeicherbereich.*

LPAP-Bündel
LPAP bundle

LPAP-Bündel ermöglichen die Verteilung von Nachrichten an LPAP-Partner auf mehrere Partner-Anwendungen. Soll eine UTM-Anwendung sehr viele Nachrichten mit einer Partner-Anwendung austauschen, kann es für die Lastverteilung sinnvoll sein, mehrere Instanzen der Partner-Anwendung zu starten und die Nachrichten auf die einzelnen Instanzen zu verteilen. In einem LPAP-Bündel übernimmt openUTM die Verteilung der Nachrichten an die Instanzen der Partner-Anwendung. Ein LPAP-Bündel besteht aus einem Master-LPAP und mehreren Slave-LPAPs. Die Slave-LPAPs werden dem Master-LPAP bei der UTM-Generierung zugeordnet. LPAP-Bündel gibt es sowohl für das OSI TP-Protokoll als auch für das LU6.1-Protokoll.

LPAP-Partner
LPAP partner

Für die *verteilte Verarbeitung* über das *LU6.1*-Protokoll muss in der lokalen Anwendung für jede Partner-Anwendung ein LPAP-Partner konfiguriert werden. Der LPAP-Partner spiegelt in der lokalen Anwendung die Partner-Anwendung wider. Bei der Kommunikation wird die Partner-Anwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten LPAP-Partners angesprochen.

LTERM-Bündel
LTERM bundle

Ein LTERM-Bündel (Verbindungsbündel) besteht aus einem Master-LTERM und mehreren Slave-LTERMs. Mit einem LTERM-Bündel (Verbindungsbündel) verteilen Sie asynchrone Nachrichten an eine logische Partner-Anwendung gleichmäßig auf mehrere parallele Verbindungen.

LTERM-Gruppe
LTERM group

Eine LTERM-Gruppe besteht aus einem oder mehreren Alias-LTERMs, den Gruppen-LTERMs, und einem Primary-LTERM. In einer LTERM-Gruppe ordnen Sie mehrere LTERMs einer Verbindung zu.

LTERM-Partner
LTERM partner

Um *Clients* oder Drucker an eine *UTM-Anwendung* anschließen zu können, müssen in der Anwendung LTERM-Partner konfiguriert werden. Ein Client oder Drucker kann nur angeschlossen werden, wenn ihm ein LTERM-Partner mit entsprechenden Eigenschaften zugeordnet ist. Diese Zuordnung wird i.A. in der *Konfiguration* festgelegt, sie kann aber auch dynamisch über Terminal-Pools erfolgen.

LTERM-Pool
LTERM pool

Statt für jeden *Client* eine LTERM- und eine PTERM-Anweisung anzugeben, kann mit der Anweisung TPOOL ein Pool von LTERM-Partnern definiert werden. Schließt sich ein Client über einen LTERM-Pool an, wird ihm dynamisch ein LTERM-Partner aus dem Pool zugeordnet.

LU6.1

Geräteunabhängiges Datenaustauschprotokoll (Industrie-Standard) für die transaktionsgesicherte *Server-Server-Kommunikation*.

LU6.1-LPAP-Bündel
LU6.1-LPAP bundle

LPAP-Bündel für *LU6.1*-Partner-Anwendungen.

LU6.1-Partner
LU6.1 partner

Partner der *UTM-Anwendung*, der mit der UTM-Anwendung über das Protokoll *LU6.1* kommuniziert. Beispiele für solche Partner sind:

- eine UTM-Anwendung, die über LU6.1 kommuniziert
- eine Anwendung im IBM-Umfeld (z.B. CICS, IMS oder TXSeries), die über LU6.1 kommuniziert

Mainprozess (Unix-, Linux- und Windows-Systeme)
main process

Prozess, der die *UTM-Anwendung* startet. Er startet die *Workprozesse*, die *UTM-System-Prozesse*, *Druckerprozesse, Netzprozesse, Logging-Prozess* und den *Timerprozess* und überwacht die *UTM-Anwendung*.

Main Routine KDCROOT
main routine KDCROOT

siehe *KDCROOT*.

Management Unit
management unit

Komponente des *SE Servers*; ermöglicht mit Hilfe des *SE Managers* ein zentrales, web-basiertes Management aller Units eines SE Servers.

Meldung / UTM-Meldung
UTM message

Meldungen werden vom Transaktionsmonitor openUTM oder von UTM-Tools (wie z.B. *KDCDEF*) an *Meldungsziele* ausgegeben. Eine Meldung besteht aus einer Meldungsnummer und dem Meldungstext, der ggf. *Inserts* mit aktuellen Werten enthält. Je nach Meldungsziel werden entweder die gesamte Meldung oder nur Teile der Meldung (z.B. nur die Inserts) ausgegeben.

Meldungsdefinitionsdatei
message definition file

Die Meldungsdefinitionsdatei wird mit openUTM ausgeliefert und enthält standardmäßig die UTM-Meldungstexte in deutscher und englischer Sprache und die Definitionen der Meldungseigenschaften. Aufbauend auf diese Datei kann der Anwender auch eigene, individuelle Meldungsmodule erzeugen.

Meldungsziel
message destination

Ausgabemedium für eine *Meldung*. Mögliche Meldungsziele von Meldungen des Transaktionsmonitors openUTM sind z.B. Terminals, *TS-Anwendungen*, der *Event-Service* MSGTAC, die *System-Protokolldatei* SYSLOG oder *TAC-Queues, Asynchron-TACs, USER-Queues*, SYSOUT/SYSLST bzw. stderr/stdout. Meldungsziele von Meldungen der UTM-Tools sind SYSOUT /SYSLST bzw. stderr/stdout.

Mehrschritt-Transaktion
multi-step transaction

Transaktion, die aus mehr als einem *Verarbeitungsschritt* besteht.

Mehrschritt-Vorgang (KDCS)
multi-step service

Vorgang, der in mehreren *Dialog-Schritten* ausgeführt wird.

Message Queuing
message queuing

Message Queuing (MQ) ist eine Form der Kommunikation, bei der die Nachrichten (Messages) nicht unmittelbar, sondern über zwischengeschaltete *Message Queues* ausgetauscht werden. Sender und Empfänger können zeitlich und räumlich entkoppelt ablaufen. Die Übermittlung der Nachricht hängt nicht davon ab, ob gerade eine Netzverbindung besteht oder nicht. Bei openUTM gibt es *UTM-gesteuerte Queues* und *Service-gesteuerte Queues*.

Message Queue
message queue

Warteschlange, in der bestimmte Nachrichten transaktionsgesichert bis zur Weiterverarbeitung eingereiht werden. Je nachdem, wer die Weiterverarbeitung kontrolliert, unterscheidet man *Service-gesteuerte Queues* und *UTM-gesteuerte Queues*.

MSGTAC
MSGTAC

Spezieller Event-Service, der Meldungen mit dem Meldungsziel MSGTAC per Programm verarbeitet. MSGTAC ist ein Asynchron-Vorgang und wird vom Betreiber der Anwendung erstellt.

Multiplex-Verbindung (BS2000-Systeme)
multiplex connection

Spezielle Möglichkeit, die *OMNIS* bietet, um Terminals an eine *UTM-Anwendung* anzuschließen. Eine Multiplex-Verbindung ermöglicht es, dass sich mehrere Terminals eine *Transportverbindung* teilen.

Nachrichten-Bereich/NB (KDCS)
KDCS message area

Bei KDCS-Aufrufen: Puffer-Bereich, in dem Nachrichten oder Daten für openUTM oder für das *Teilprogramm* bereitgestellt werden.

Network File System/Service / NFS

Ermöglicht den Zugriff von Unix- und Linux-Rechnern auf Dateisysteme über das Netzwerk.

Netzprozess (Unix-, Linux- und Windows-Systeme)
net process

Prozess einer *UTM-Anwendung* zur Netzanbindung.

Netzwerk-Selektor
network selector

Der Netzwerk-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Vermittlungsschicht des *OSI-Referenzmodells*.

Normale Beendigung einer UTM-Anwendung
normal termination of a UTM application

Kontrollierte Beendigung einer *UTM-Anwendung*; das bedeutet u.a., dass die Verwaltungsdaten auf der *KDCFILE* aktualisiert werden. Eine normale Beendigung veranlasst der *Administrator* (z.B. mit KDCSHUT N). Den Start nach einer normalen Beendigung führt openUTM als *Kaltstart* durch.

Object Identifier
object identifier

Ein Object Identifier ist ein weltweit eindeutiger Bezeichner für Objekte im OSI-Umfeld. Ein Object Identifier besteht aus einer Folge von ganzen Zahlen, die einen Pfad in einer Baumstruktur repräsentiert.

Offener Terminalpool
open terminal pool

Terminalpool, der nicht auf *Clients* eines Rechners oder eines bestimmten Typs beschränkt ist. An diesen Terminalpool können sich alle Clients anschließen, für die kein Rechner- oder Typ-spezifischer Terminalpool generiert ist.

OMNIS (BS2000-Systeme)
OMNIS

OMNIS ist ein „Session-Manager“ auf einem BS2000-System, der die gleichzeitige Verbindungsaufnahme von einem Terminal zu mehreren Partnern in einem Netzwerk ermöglicht. OMNIS ermöglicht es außerdem, mit *Multiplex-Verbindungen* zu arbeiten.

Online-Import
online import

Als Online-Import wird in einer *UTM-Cluster-Anwendung* das Importieren von Anwendungsdaten aus einer normal beendeten Knoten-Anwendung in eine laufende Knoten-Anwendung bezeichnet.

Online-Update
online update

Als Online-Update wird in einer *UTM-Cluster-Anwendung* die Änderung der Konfiguration der Anwendung oder des Anwendungsprogramms oder der Einsatz einer neuen UTM-Korrekturstufe bei laufender *UTM-Cluster-Anwendung* bezeichnet.

OpenCPIC

Trägersystem für UTM-Clients, die das *OSI TP* Protokoll verwenden.

OpenCPIC-Client
OpenCPIC client

OSI TP Partner-Anwendungen mit Trägersystem *OpenCPIC*.

openSM2

Die Produktlinie openSM2 ist eine einheitliche Lösung für das unternehmensweite Performance Management von Server- und Speichersystemen. openSM2 bietet eine Messdatenerfassung, Online-Überwachung und Offline-Auswertung.

openUTM-Cluster
openUTM cluster

aus der Sicht von UPIC-Clients, **nicht** aus Server-Sicht:
Zusammenfassung mehrerer Knoten-Anwendungen einer UTM-Cluster-Anwendung zu einer logischen Anwendung, die über einen gemeinsamen Symbolic Destination Name adressiert wird.

openUTM-D

openUTM-D (openUTM-Distributed) ist eine openUTM-Komponente, die *verteilte Verarbeitung* ermöglicht. openUTM-D ist integraler Bestandteil von openUTM.

OSI-LPAP-Bündel
OSI-LPAP bundle

LPAP-Bündel für *OSI TP*-Partner-Anwendungen.

OSI-LPAP-Partner
OSI-LPAP partner

OSI-LPAP-Partner sind die bei openUTM generierten Adressen der *OSI TP-Partner*. Für die *verteilte Verarbeitung* über das Protokoll *OSI TP* muss in der lokalen Anwendung für jede Partner-Anwendung ein OSI-LPAP-Partner konfiguriert werden. Der OSI-LPAP-Partner spiegelt in der lokalen Anwendung die Partner-Anwendung wider. Bei der Kommunikation wird die Partner-Anwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten OSI-LPAP-Partners angesprochen.

OSI-Referenzmodell
OSI reference model

Das OSI-Referenzmodell stellt einen Rahmen für die Standardisierung der Kommunikation von offenen Systemen dar. ISO, die Internationale Organisation für Standardisierung, hat dieses Modell im internationalen Standard
ISO IS7498 beschrieben. Das OSI-Referenzmodell unterteilt die für die Kommunikation von Systemen notwendigen Funktionen in sieben logische Schichten. Diese Schichten haben jeweils klar definierte Schnittstellen zu den benachbarten Schichten.

OSI TP

Von der ISO definiertes Kommunikationsprotokoll für die verteilte Transaktionsverarbeitung. OSI TP steht für Open System Interconnection Transaction Processing.

OSI TP-Partner
OSI TP partner

Partner der UTM-Anwendung, der mit der UTM-Anwendung über das OSI TP-Protokoll kommuniziert.
Beispiele für solche Partner sind:

- eine UTM-Anwendung, die über OSI TP kommuniziert
- eine Anwendung im IBM-Umfeld (z.B. CICS), die über openUTM-LU62 angeschlossen ist
- ein *OpenCPIC-Client*
- Anwendungen anderer TP-Monitore, die OSI TP unterstützen

Outbound-Conversation (CPI-C)
outbound conversation

siehe *Outgoing-Conversation*.

Outgoing-Conversation (CPI-C)
outgoing conversation

Eine Conversation, bei der das lokale CPI-C-Programm der *Initiator* ist, heißt Outgoing-Conversation. In der X/Open-Specification wird für Outgoing-Conversation auch das Synonym Outbound-Conversation verwendet.

Pagepool
page pool

Teil der *KDCFILE*, in dem Anwenderdaten gespeichert werden.
In einer *stand-alone Anwendung* sind dies z.B. *Dialog-Nachrichten*, Nachrichten an *Message Queues*, *Sekundärspeicherbereiche*.
In einer *UTM-Cluster-Anwendung* sind dies z.B. Nachrichten an *Message Queues, TLS*.

Parameterbereich
parameter area

Datenstruktur, in der ein *Teilprogramm* bei einem UTM-Aufruf die für diesen Aufruf notwendigen Operanden an openUTM übergibt.

Partner-Anwendung
partner application

Partner einer UTM-Anwendung bei *verteilter Verarbeitung*. Für die verteilte Verarbeitung werden höhere Kommunikationsprotokolle verwendet (*LU6.1*, *OSI TP* oder LU6.2 über das Gateway openUTM-LU62).

Postselection (BS2000-Systeme)
postselection

Auswahl der protokollierten UTM-Ereignisse aus der SAT-Protokolldatei, die ausgewertet werden sollen. Die Auswahl erfolgt mit Hilfe des Tools SATUT.

Programmraum (BS2000-Systeme)
program space

In Speicherklassen aufgeteilter virtueller Adressraum des BS2000, in dem sowohl ablauffähige Programme als auch reine Daten adressiert werden.

Prepare to commit (PTC)
prepare to commit

Bestimmter Zustand einer verteilten Transaktion:
Das Transaktionsende der verteilten Transaktion wurde eingeleitet, es wird jedoch noch auf die Bestätigung des Transaktionsendes durch den Partner gewartet.

Preselection (BS2000-Systeme)
preselection

Festlegung der für die *SAT-Beweissicherung* zu protokollierenden UTM-Ereignisse. Die Preselection erfolgt durch die UTM-SAT-Administration. Man unterscheidet Ereignis-spezifische, Benutzer-spezifische und Auftrags-(TAC-)spezifische Preselection.

Presentation-Selektor
presentation selector

Der Presentation-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Darstellungsschicht des *OSI-Referenzmodells*.

Primärspeicherbereich
primary storage area

Bereich im Arbeitsspeicher, auf den das *KDCS-Teilprogramm* direkt zugreifen kann, z.B. *Standard Primärer Arbeitsbereich*, *Kommunikationsbereich*.

Printerprozess (Unix- und Linux-Systeme)
printer process

siehe *Druckerprozess*.

Programmschnittstelle zur Administration
program interface for administration

UTM-Programmschnittstelle, mit deren Hilfe der Anwender eigene *Administrationsprogramme* erstellen kann. Die Programmschnittstelle zur Administration bietet u.a. Funktionen zur *dynamischen Konfiguration*, zur Modifikation von Eigenschaften und Anwendungsparametern und zur Abfrage von Informationen zur *Konfiguration* und zur aktuellen Auslastung der Anwendung.

Prozess
prozess

In den openUTM-Handbüchern wird der Begriff "Prozess" als Oberbegriff für Prozess (Unix-, Linux- und Windows-Systeme) und Task (BS2000-Systeme) verwendet.

Queue
queue

siehe *Message Queue*

Quick Start Kit

Beispielanwendung, die mit openUTM (Windows-Systeme) ausgeliefert wird.

Quittungs-Auftrag
confirmation job

Bestandteil eines *Auftrags-Komplexes*, worin der Quittungs-Auftrag dem *Basis-Auftrag* zugeordnet ist. Es gibt positive und negative Quittungsaufträge. Bei positivem Ergebnis des *Basis-Auftrags* wird der positive Quittungs-Auftrag wirksam, sonst der negative.

Redelivery
redelivery

Erneutes Zustellen einer *Asynchron-Nachricht*, nachdem diese nicht ordnungsgemäß verarbeitet werden konnte, z.B. weil die *Transaktion* zurückgesetzt oder der *Asynchron-Vorgang* abnormal beendet wurde. Die Nachricht wird wieder in die Message Queue eingereiht und lässt sich damit erneut lesen und/oder verarbeiten.

Reentrant-fähiges Programm
reentrant program

Programm, dessen Code durch die Ausführung nicht verändert wird.
Auf BS2000-Systemen ist dies Voraussetzung dafür, *Shared Code* zu nutzen.

Request
request

Anforderung einer *Service-Funktion* durch einen *Client* oder einen anderen Server.

Requestor
requestor

In XATMI steht der Begriff Requestor für eine Anwendung, die einen Service aufruft.

Resource Manager
resource manager

Resource Manager (RMs) verwalten Datenressourcen. Ein Beispiel für RMs sind Datenbank-Systeme. openUTM stellt aber auch selbst Resource Manager zur Verfügung, z.B. für den Zugriff auf *Message Queues*, lokale Speicherbereiche und Logging-Dateien. Anwendungsprogramme greifen auf RMs über RM-spezifische Schnittstellen zu. Für Datenbank-Systeme ist dies meist SQL, für die openUTM-RMs die Schnittstelle KDCS.

RFC1006

Von IETF (Internet Engineering Task Force) definiertes Protokoll der TCP/IP-Familie zur Realisierung der ISO-Transportdienste (Transportklasse 0) auf TCP/IP-Basis.

RSA

Abkürzung für die Erfinder des RSA-Verschlüsselungsverfahrens Rivest, Shamir und Adleman. Bei diesem Verfahren wird ein Schlüsselpaar verwendet, das aus einem öffentlichen und einem privaten Schlüssel besteht. Eine Nachricht wird mit dem öffentlichen Schlüssel verschlüsselt und kann nur mit dem privaten Schlüssel entschlüsselt werden. Das RSA-Schlüsselpaar wird von der UTM-Anwendung erzeugt.

SAT-Beweissicherung (BS2000-Systeme)
SAT audit

Beweissicherung durch die Komponente SAT (Security Audit Trail) des BS2000-Softwareproduktes SECOS.

SE Manager
SE manager

Web-basierte Benutzeroberfläche (GUI) für Business Server der SE Serie. Der SE Manager läuft auf der *Management Unit* und ermöglicht die zentrale Bedienung und Verwaltung von Server Units (mit /390-Architektur und/oder x86-Architektur), Application Units (x86-Architektur), Net Unit und der Peripherie.

SE Server
SE server

Ein Business Server der SE Serie von Fujitsu.

Sekundärspeicherbereich
secondary storage area

Transaktionsgesicherter Speicherbereich, auf den das KDCS-*Teilprogramm* mit speziellen Aufrufen zugreifen kann. Lokale Sekundärspeicherbereiche (LSSB) sind einem *Vorgang* zugeordnet, auf globale Sekundärspeicherbereiche (GSSB) kann von allen Vorgängen einer *UTM-Anwendung* zugegriffen werden. Weitere Sekundärspeicherbereiche sind der *Terminal-spezifische Langzeitspeicher (TLS)* und der *User-spezifische Langzeitspeicher (ULS)* .

Selektor
selector

Ein Selektor identifiziert im lokalen System einen *Zugriffspunkt* auf die Dienste einer Schicht des *OSI-Referenzmodells*. Jeder Selektor ist Bestandteil der Adresse des Zugriffspunktes.

Semaphor (Unix-, Linux- und Windows-Systeme)
semaphore

Betriebsmittel auf Unix-, Linux- und Windows-Systemen, das zur Steuerung und Synchronisation von Prozessen dient.

Server
server

Ein Server ist eine *Anwendung*, die *Services* zur Verfügung stellt. Oft bezeichnet man auch den Rechner, auf dem Anwendungen laufen, als Server.

Server-Seite einer Conversation (CPI-C)
server side of a conversation

Begriff ersetzt durch *Akzeptor*.

Server-Server-Kommunikation
server-server communication

siehe *verteilte Verarbeitung*.

Service Access Point

siehe *Dienstzugriffspunkt.*

Service
service

Services bearbeiten die Aufträge, die an eine Server-Anwendung geschickt werden. Ein Service in einer UTM-Anwendung wird auch Vorgang genannt und setzt sich aus einer oder mehreren Transaktionen zusammen. Ein Service wird über den Vorgangs-TAC aufgerufen. Services können von Clients oder anderen Services angefordert werden.

Service-gesteuerte Queue
service controlled queue

Message Queue, bei der der Abruf und die Weiterverarbeitung der Nachrichten durch Services gesteuert werden. Ein Service muss zum Lesen der Nachricht explizit einen KDCS-Aufruf (DGET) absetzen.
Service-gesteuerte Queues gibt es bei openUTM in den Varianten USER-Queue, TAC-Queue und Temporäre Queue.

Service Routine
service routine

siehe Teilprogramm.

Session
session

Kommunikationsbeziehung zweier adressierbarer Einheiten im Netz über das SNA-Protokoll LU6.1.

Session-Selektor
session selector

Der Session-Selektor identifiziert im lokalen System einen Zugriffspunkt zu den Diensten der Kommunikationssteuerschicht (Session-Layer) des OSI-Referenzmodells.

Shared Code (BS2000-Systeme)
shared code

Code, der von mehreren Prozessen gemeinsam benutzt werden kann.

Shared Memory
shared memory

Virtueller Speicherbereich, auf den mehrere Prozesse gleichzeitig zugreifen können.

Shared Objects (Unix-, Linux- und Windows-Systeme)
shared objects

Teile des Anwendungsprogramms können als Shared Objects erzeugt werden. Diese werden dynamisch zur Anwendung dazugebunden und können im laufenden Betrieb ausgetauscht werden. Shared Objects werden mit der KDCDEF-Anweisung SHARED-OBJECT definiert.

Sicherungspunkt
synchronization point, consistency point

Ende einer Transaktion. Zu diesem Zeitpunkt werden alle in der Transaktion vorgenommenen Änderungen der Anwendungsinformation gegen Systemausfall gesichert und für andere sichtbar gemacht. Während der Transaktion gesetzte Sperren werden wieder aufgehoben.

Single System Image

Unter single system image versteht man die Eigenschaft eines Clusters, nach außen hin als ein einziges, in sich geschlossenes System zu erscheinen. Die heterogene Natur des Clusters und die interne Verteilung der Ressourcen im Cluster ist für die Benutzer des Clusters und die Anwendungen, die mit dem Cluster kommunizieren, nicht sichtbar.

SOA

SOA (Service-oriented architecture).
SOA ist ein Konzept für eine Systemarchitektur, in dem Funktionen in Form von wieder verwendbaren, technisch voneinander unabhängigen und fachlich lose gekoppelten Services implementiert werden. Services können unabhängig von zugrunde liegenden Implementierungen über Schnittstellen aufgerufen werden, deren Spezifikationen öffentlich und damit vertrauenswürdig sein können. Service-Interaktion findet über eine dafür vorgesehene Kommunikationsinfrastruktur statt.

SOAP

SOAP (Simple Object Access Protocol) ist ein Protokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Remote Procedure Calls durchgeführt werden können. SOAP stützt sich auf die Dienste anderer Standards, XML zur Repräsentation der Daten und Internet-Protokolle der Transport- und Anwendungsschicht zur Übertragung der Nachrichten.

Socket-Verbindung
socket connection

Transportsystem-Verbindung, die die Socket-Schnittstelle verwendet. Die Socket-Schnittstelle ist eine Standard-Programmschnittstelle für die Kommunikation über TCP/IP.

Stand-alone Anwendung
stand-alone application

siehe stand-alone UTM-Anwendung.

Stand-alone UTM-Anwendung
stand-alone UTM application

Herkömmliche UTM-Anwendung, die nicht Bestandteil einer UTM-Cluster-Anwendung ist.

Standard Primärer Arbeitsbereich/SPAB (KDCS)
standard primary working area

Bereich im Arbeitsspeicher, der jedem KDCS-Teilprogramm zur Verfügung steht. Sein Inhalt ist zu Beginn des Teilprogrammlaufs undefiniert oder mit einem Füllzeichen vorbelegt.

Startformat
start format

Format, das openUTM am Terminal ausgibt, wenn sich ein Benutzer erfolgreich bei der UTM-Anwendung angemeldet hat (ausgenommen nach Vorgangs-Wiederanlauf und beim Anmelden über Anmelde-Vorgang).

Statische Konfiguration
static configuration

Festlegen der Konfiguration bei der UTM-Generierung mit Hilfe des UTM-Tools KDCDEF.

SYSLOG-Datei
SYSLOG file

siehe System-Protokolldatei.

System-Protokolldatei
system log file

Datei oder Dateigeneration, in die openUTM während des Laufs einer UTM-Anwendung alle UTM-Meldungen protokolliert, für die das Meldungsziel SYSLOG definiert ist.

TAC
TAC

siehe Transaktionscode.

TAC-Queue
TAC queue

Message Queue, die explizit per KDCDEF-Anweisung generiert wird. Eine TAC-Queue ist eine Service-gesteuerte Queue und kann unter dem generierten Namen von jedem Service aus angesprochen werden.

Teilprogramm
program unit

UTM-Services werden durch ein oder mehrere Teilprogramme realisiert. Die Teilprogramme sind Bestandteile des Anwendungsprogramms. Abhängig vom verwendeten API müssen sie KDCS-, XATMI- oder CPIC-Aufrufe enthalten. Sie sind über Transaktionscodes ansprechbar. Einem Teilprogramm können mehrere Transaktionscodes zugeordnet werden.

Temporäre Queue
temporary queue

Message Queue, die dynamisch per Programm erzeugt wird und auch wieder per Programm gelöscht werden kann, vgl. Service-gesteuerte Queue.

Terminal-spezifischer Langzeitspeicher/TLS (KDCS)
terminal-specific long-term storage

Sekundärspeicher, der einem LTERM-, LPAP- oder OSI-LPAP-Partner zugeordnet ist und über das Anwendungsende hinaus erhalten bleibt.

Timerprozess (Unix-, Linux- und Windows-Systeme)
timer process

Prozess, der Aufträge zur Zeitüberwachung von Workprozessen entgegennimmt, sie in ein Auftragsbuch einordnet und nach einer im Auftragsbuch festgelegten Zeit den Workprozessen zur Bearbeitung wieder zustellt.

TLS Termination Proxy
TLS termination proxy

Ein TLS-Terminierungsproxy ist ein Proxy-Server, der verwendet wird, um eingehende TLS-Verbindungen zu verarbeiten, die Daten zu entschlüsseln und die unverschlüsselte Anforderung an andere Server weiterzugeben.

TNS (Unix-, Linux- und Windows-Systeme)

Abkürzung für den Transport Name Service, der einem Anwendungsnamen einen Transport-Selektor und das Transportsystem zuordnet, über das die Anwendung erreichbar ist.

Tomcat

siehe Apache Tomcat

Transaktion
transaction

Verarbeitungsabschnitt innerhalb eines Services, für den die Einhaltung der ACID-Eigenschaften garantiert wird. Von den in einer Transaktion beabsichtigten Änderungen der Anwendungsinformation werden entweder alle konsistent durchgeführt oder es wird keine durchgeführt (Alles-oder-Nichts Regel). Das Transaktionsende bildet einen Sicherungspunkt.

Transaktionscode/TAC
transaction code

Name, über den ein Teilprogramm aufgerufen werden kann. Der Transaktionscode wird dem Teilprogramm bei der statischen oder dynamischen Konfiguration zugeordnet. Einem Teilprogramm können auch mehrere Transaktionscodes zugeordnet werden.

Transaktionsrate
transaction rate

Anzahl der erfolgreich beendeten Transaktionen pro Zeiteinheit.

Transfer-Syntax
transfer syntax

Bei OSI TP werden die Daten zur Übertragung zwischen zwei Rechnersystemen von der lokalen Darstellung in die Transfer-Syntax umgewandelt. Die Transfer-Syntax beschreibt die Daten in einem neutralen Format, das von allen beteiligten Partnern verstanden wird. Jeder Transfer-Syntax muss ein Object Identifier zugeordnet sein.

Transport Layer Security
transport layer security

Der Transport Layer Security, ist ein hybrides Verschlüsselungsprotokoll zur sicheren Datenübertragung im Internet.

Transport-Selektor
transport selector

Der Transport-Selektor identifiziert im lokalen System einen Dienstzugriffspunkt zur Transportschicht des OSI-Referenzmodells.

Transportsystem-Anwendung
transport system application

Anwendung, die direkt auf einer Transportsystem-Schnittstelle wie z.B. CMX, DCAM oder Socket aufsetzt. Für den Anschluss von Transportsystem-Anwendungen muss bei der Konfiguration als Partnertyp APPLI oder SOCKET angegeben werden. Eine Transportsystem-Anwendung kann nicht in eine Verteilte Transaktion eingebunden werden.

Transportsystem-Endpunkt
transport system end point

Bei der Client-/Server- oder Server-/Server-Kommunikation wird eine Verbindung zwischen zwei Transportsystem-Endpunkten aufgebaut. Ein Transportsystem-Endpunkt wird auch als lokaler Anwendungsname bezeichnet und wird mit der Anweisung BCAMAPPL oder mit MAX APPLINAME definiert.

Transportsystem-Zugriffspunkt
transport system access point

siehe Transportsystem-Endpunkt.

Transportverbindung
transport connection

Im OSI-Referenzmodell eine Verbindung zwischen zwei Instanzen der Schicht 4 (Transportschicht).

TS-Anwendung
TS application

siehe Transportsystem-Anwendung.

Typisierter Puffer (XATMI)
typed buffer

Puffer für den Austausch von typisierten und strukturierten Daten zwischen Kommunikationspartnern. Durch diese typisierten Puffer ist die Struktur der ausgetauschten Daten den Partnern implizit bekannt.

UPIC

Trägersystem für UTM-Clients. UPIC steht für Universal Programming Interface for Communication. Die Kommunikation mit der UTM-Anwendung erfolgt über das UPIC-Protokoll.

UPIC-Client

Bezeichnung für UTM-Clients mit Trägersystem UPIC und JConnect-Clients.

UPIC-Protokoll
Upic protocol

Protokoll für die Client-Server-Kommunikation mit UTM-Anwendungen. Das UPIC-Protokoll wird von UPIC-Clients und von JConnect-Clients verwendet.

UPIC Analyzer

Komponente zur Analyse der mit UPIC Capture mitgeschnittenen UPIC-Kommunikation. Dieser Schritt dient dazu, den Mitschnitt für das Abspielen mit UPIC Replay aufzubereiten.

UPIC Capture

Mitschneiden der Kommunikation zwischen UPIC-Clients und UTM-Anwendungen, um sie zu einem späteren Zeitpunkt abspielen zu können
(UPIC Replay).

UPIC Replay

Komponente zum Abspielen der mit UPIC Capture mitgeschnittenen und mit UPIC Analyzer aufbereiteten UPIC-Kommunikation.

USER-Queue
USER queue

Message Queue, die openUTM jeder Benutzerkennung zur Verfügung stellt. Eine USER-Queue zählt zu den Service-gesteuerten Queues und ist immer der jeweiligen Benutzerkennung zugeordnet. Der Zugriff von fremden UTM-Benutzern auf die eigene USER-Queue kann eingeschränkt werden.

User-spezifischer Langzeitspeicher/ULS
user-specific long-term storage

Sekundärspeicher, der einer Benutzerkennung, einer Session oder einer Association zugeordnet ist und über das Anwendungsende hinaus erhalten bleibt.

USLOG-Datei
USLOG file

siehe Benutzer-Protokolldatei.

UTM-Anwendung
UTM application

Eine UTM-Anwendung stellt Services zur Verfügung, die Aufträge von Clients oder anderen Anwendungen bearbeiten. openUTM übernimmt dabei u.a. die Transaktionssicherung und das Management der Kommunikations- und Systemressourcen. Technisch gesehen ist eine UTM-Anwendung eine Prozessgruppe, die zur Laufzeit eine logische Server-Einheit bildet.

UTM-Client
UTM client

siehe Client.

UTM-Cluster-Anwendung
UTM cluster application

UTM-Anwendung, die für den Einsatz in einem Cluster generiert ist und die man logisch als **eine** Anwendung betrachten kann.
Physikalisch gesehen besteht eine UTM-Cluster-Anwendung aus mehreren, identisch generierten UTM-Anwendungen, die auf den einzelnen Knoten laufen.

UTM-Cluster-Dateien
UTM cluster files

Oberbegriff für alle Dateien, die für den Ablauf einer UTM-Cluster-Anwendung auf Unix-, Linux- und Windows-Systemen benötigt werden. Dazu gehören folgende Dateien:

- Cluster-Konfigurationsdatei
- Cluster-User-Datei
- Dateien des Cluster-Pagepool
- Cluster-GSSB-Datei
- Cluster-ULS-Datei
- Dateien des Cluster-Administrations-Journals*
- Cluster-Lock-Datei*
- Lock-Datei zur Start-Serialisierung*

Die mit * gekennzeichneten Dateien werden beim Start der ersten Knoten-Anwendung angelegt, alle anderen Dateien werden bei der Generierung mit KDCDEF erzeugt.

UTM-D

siehe openUTM-D.

UTM-Datenstation
UTM terminal

Begriff ersetzt durch LTERM-Partner.

UTM-F

UTM-Anwendungen können als UTM-F-Anwendungen (UTM-Fast) generiert werden. Bei UTM-F wird zugunsten der Performance auf Platteneingaben/-ausgaben verzichtet, mit denen bei UTM-S die Sicherung von Benutzer- und Transaktionsdaten durchgeführt wird. Gesichert werden lediglich Änderungen der Verwaltungsdaten.
In UTM-Cluster-Anwendungen, die als UTM-F-Anwendung generiert sind (APPLIMODE=FAST), werden Cluster-weit gültige Anwenderdaten auch gesichert. Dabei werden GSSB- und ULS-Daten genauso behandelt wie in UTM-Cluster-Anwendungen, die mit UTM-S generiert sind. Vorgangs-Daten von Benutzern mit RESTART=YES werden jedoch nur beim Abmelden des Benutzers anstatt bei jedem Transaktionsende geschrieben.

UTM-Generierung
UTM generation

Statische Konfiguration einer UTM-Anwendung mit dem UTM-Tool KDCDEF und Erzeugen des Anwendungsprogramms.

UTM-gesteuerte Queues
UTM controlled queue

Message Queues, bei denen der Abruf und die Weiterverarbeitung der Nachrichten vollständig durch openUTM gesteuert werden. Siehe auch Asynchron-Auftrag, Hintergrund-Auftrag und Asynchron-Nachricht.

UTM-S

Bei UTM-S-Anwendungen sichert openUTM neben den Verwaltungsdaten auch alle Benutzerdaten über ein Anwendungsende und einen Systemausfall hinaus. Außerdem garantiert UTM-S bei allen Störungen die Sicherheit und Konsistenz der Anwendungsdaten. Im Standardfall werden UTM-Anwendungen als UTM-S-Anwendungen (UTM-Secure) generiert.

UTM-SAT-Administration (BS2000-Systeme)
UTM SAT administration

Durch die UTM-SAT-Administration wird gesteuert, welche sicherheitsrelevanten UTM-Ereignisse, die im Betrieb der UTM-Anwendung auftreten, von SAT protokolliert werden sollen. Für die UTM-SAT-Administration wird eine besondere Berechtigung benötigt.

UTM-Seite
UTM page

Ist eine Speichereinheit, die entweder 2K, 4K oder 8K umfasst. In stand-alone UTM-Anwendungen kann die Größe einer UTM-Seite bei der Generierung der UTM-Anwendung auf 2K, 4K oder 8K gesetzt werden. In einer UTM-Cluster-Anwendung ist die Größe einer UTM-Seite immer 4K oder 8K. Pagepool und Wiederanlauf-Bereich der KDCFILE sowie UTM-Cluster-Dateien werden in Einheiten der Größe einer UTM-Seite unterteilt.

UTM Socket Protokoll (USP)
UTM socket protocol

Proprietäres Protokoll von openUTM oberhalb von TCP/IP zur Umsetzung der über die Socket-Schnittstelle empfangenen Bytestreams in Nachrichten.

UTM-System-Prozess
UTM system process

UTM-Prozess, der zusätzlich zu den per Startparameter angegebenen Prozessen gestartet wird und nur ausgewählte Aufträge bearbeitet. UTM-System-Prozesse dienen dazu, eine UTM-Anwendung auch bei sehr hoher Last reaktionsfähig zu halten.

UTM-Tool
UTM tool

Programm, das zusammen mit openUTM zur Verfügung gestellt und für bestimmte UTM-spezifische Aufgaben benötigt wird (z.B. zum Konfigurieren).

utmpfad (Unix-, Linux- und Windows-Systeme)
utmpath

Das Dateiverzeichnis unter dem die Komponenten von openUTM installiert sind, wird in diesem Handbuch als utmpfad bezeichnet.
Um einen korrekten Ablauf von openUTM zu garantieren, muss die Umgebungsvariable UTMPATH auf den Wert von utmpfad gesetzt werden. Auf Unix- und Linux-Systemen müssen Sie UTMPATH vor dem Starten einer UTM-Anwendung setzen. Auf Windows-Systemen wird UTMPATH passend zu der zuletzt installierten UTM-Version gesetzt.

Verarbeitungsschritt
processing step

Ein Verarbeitungsschritt beginnt mit dem Empfangen einer Dialog-Nachricht, die von einem Client oder einer anderen Server-Anwendung an die UTM-Anwendung gesendet wird. Der Verarbeitungsschritt endet entweder mit dem Senden einer Antwort und beendet damit auch den Dialog-Schritt oder er endet mit dem Senden einer Dialog-Nachricht an einen Dritten.

Verbindungs-Benutzerkennung
connection user ID

Benutzerkennung, unter der eine TS-Anwendung oder ein UPIC-Client direkt nach dem Verbindungsaufbau bei der UTM-Anwendung angemeldet wird. Abhängig von der Generierung des Clients (= LTERM-Partner) gilt:

- Die Verbindungs-Benutzerkennung ist gleich dem USER der LTERM-Anweisung (explizite Verbindungs-Benutzerkennung). Eine explizite Verbindungs-Benutzerkennung muss mit einer USER-Anweisung generiert sein und kann nicht als "echte" Benutzerkennung verwendet werden.
- Die Verbindungs-Benutzerkennung ist gleich dem LTERM-Partner (implizite Verbindungs-Benutzerkennung), wenn bei der LTERM-Anweisung kein USER angegeben wurde oder wenn ein LTERM-Pool generiert wurde.

In einer UTM-Cluster-Anwendung ist der Vorgang einer Verbindungs-Benutzerkennung (RESTART=YES bei LTERM oder USER) an die Verbindung gebunden und damit Knoten-lokal. Eine Verbindungs-Benutzerkennung, die mit RESTART=YES generiert ist, kann in jeder Knoten-Anwendung einen eigenen Vorgang haben.

Verbindungsbündel
connection bundle

siehe LTERM-Bündel.

Verschlüsselungsstufe
encryption level

Die Verschlüsselungsstufe legt fest, ob und inwieweit ein Client Nachrichten und Passwort verschlüsseln muss.

Verteilte Transaktion
distributed transaction

Transaktion, die sich über mehr als eine Anwendung erstreckt und in mehreren (Teil-)Transaktionen in verteilten Systemen ausgeführt wird.

Verteilte Transaktionsverarbeitung
Distributed Transaction Processing

Verteilte Verarbeitung mit verteilten Transaktionen.

Verteilte Verarbeitung
distributed processing

Bearbeitung von Dialog-Aufträgen durch mehrere Anwendungen oder Übermittlung von Hintergrundaufträgen an eine andere Anwendung. Für die verteilte Verarbeitung werden die höheren Kommunikationsprotokolle LU6.1 und OSI TP verwendet. Über openUTM-LU62 ist verteilte Verarbeitung auch mit LU6.2 Partnern möglich. Man unterscheidet verteilte Verarbeitung mit verteilten Transaktionen (Anwendungs-übergreifende Transaktionssicherung) und verteilte Verarbeitung ohne verteilte Transaktionen (nur lokale Transaktionssicherung). Die verteilte Verarbeitung wird auch Server-Server-Kommunikation genannt.

Vorgang (KDCS)
service

Ein Vorgang dient zur Bearbeitung eines Auftrags in einer UTM-Anwendung. Er setzt sich aus einer oder mehreren Transaktionen zusammen. Die erste Transaktion wird über den Vorgangs-TAC aufgerufen. Es gibt Dialog-Vorgänge und Asynchron-Vorgänge. openUTM stellt den Teilprogrammen eines Vorgangs gemeinsame Datenbereiche zur Verfügung. Anstelle des Begriffs Vorgang wird häufig auch der allgemeinere Begriff Service gebraucht.

Vorgangs-Kellerung (KDCS)
service stacking

Ein Terminal-Benutzer kann einen laufenden Dialog-Vorgang unterbrechen und einen neuen Dialog-Vorgang einschieben. Nach Beendigung des eingeschobenen Vorgangs wird der unterbrochene Vorgang fortgesetzt.

Vorgangs-Kettung (KDCS)
service chaining

Bei Vorgangs-Kettung wird nach Beendigung eines Dialog-Vorgangs ohne Angabe einer Dialog-Nachricht ein Folgevorgang gestartet.

Vorgangs-TAC (KDCS)
service TAC

Transaktionscode, mit dem ein Vorgang gestartet wird.

Vorgangs-Wiederanlauf (KDCS)
service restart

Wird ein Vorgang unterbrochen, z.B. infolge Abmeldens des Terminal-Benutzers oder Beendigung der UTM-Anwendung, führt openUTM einen Vorgangs-Wiederanlauf durch. Ein Asynchron-Vorgang wird neu gestartet oder beim zuletzt erreichten Sicherungspunkt fortgesetzt, ein Dialog-Vorgang wird beim zuletzt erreichten Sicherungspunkt fortgesetzt. Für den Terminal-Benutzer wird der Vorgangs-Wiederanlauf eines Dialog-Vorgangs als Bildschirm-Wiederanlauf sichtbar, sofern am letzten Sicherungspunkt eine Dialog-Nachricht an den Terminal-Benutzer gesendet wurde.

Warmstart
warm start

Start einer UTM-S-Anwendung nach einer vorhergehenden abnormalen Beendigung. Dabei wird die Anwendungsinformation auf den zuletzt erreichten konsistenten Zustand gesetzt. Unterbrochene Dialog-Vorgänge werden dabei auf den zuletzt erreichten Sicherungspunkt zurückgesetzt, so dass die Verarbeitung an dieser Stelle wieder konsistent aufgenommen werden kann (Vorgangs-Wiederanlauf). Unterbrochene Asynchron-Vorgänge werden zurückgesetzt und neu gestartet oder beim zuletzt erreichten Sicherungspunkt fortgesetzt.
Bei UTM-F-Anwendungen werden beim Start nach einer vorhergehenden abnormalen Beendigung lediglich die dynamisch geänderten Konfigurationsdaten auf den zuletzt erreichten konsistenten Zustand gesetzt.
In UTM-Cluster-Anwendungen werden die globalen Sperren auf GSSB und ULS, die bei der abnormalen Beendigung von dieser Knoten-Anwendung gehalten wurden, aufgehoben. Außerdem werden Benutzer, die zum Zeitpunkt der abnormalen Beendigung an dieser Knoten-Anwendung angemeldet waren, abgemeldet.

Web Service
web service

Anwendung, die auf einem Web-Server läuft und über eine standardisierte und programmatische Schnittstelle (öffentlich) verfügbar ist. Die Web Services-Technologie ermöglicht es, UTM-Teilprogramme für moderne Web-Client-Anwendungen verfügbar zu machen, unabhängig davon, in welcher Programmiersprache sie entwickelt wurden.

WebAdmin
WebAdmin

Web-basiertes Tool zur Administration von openUTM-Anwendungen über Web-Browser. WebAdmin enthält neben dem kompletten Funktionsumfang der Programmschnittstelle zur Administration noch zusätzliche Funktionen.

Wiederanlauf
restart

siehe Bildschirm-Wiederanlauf,
siehe Vorgangs-Wiederanlauf.

WinAdmin
WinAdmin

Java-basiertes Tool zur Administration von openUTM-Anwendungen über eine grafische Oberfläche. WinAdmin enthält neben dem kompletten Funktionsumfang der Programmschnittstelle zur Administration noch zusätzliche Funktionen.

Workload Capture & Replay
workload capture & replay

Programmfamilie zur Simulation von Lastsituationen, bestehend aus den Haupt-Komponenten UPIC Capture, UPIC Analyzer und Upic Replay und auf Unix-, Linux- und Windows-Systemen dem Dienstprogramm kdcsort. Mit Workload Capture & Replay lassen sich UPIC-Sessions mit UTM-Anwendungen aufzeichnen, analysieren und mit veränderten Lastparametern wieder abspielen.

Workprozess (Unix-, Linux- und Windows-Systeme)
work process

Prozess, in dem die Services der UTM-Anwendung ablaufen.

WS4UTM

WS4UTM (**W**eb**S**ervices for open**UTM**) ermöglicht es Ihnen, auf komfortable Weise einen Service einer UTM-Anwendung als Web Service zur Verfügung zu stellen.

XATMI

XATMI (X/Open Application Transaction Manager Interface) ist eine von X/Open standardisierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen.
Das in openUTM implementierte XATMI genügt der XATMI CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung. XATMI in openUTM kann über die Protokolle OSI TP, LU6.1 und UPIC kommunizieren.

XHCS (BS2000-Systeme)

XHCS (Extended Host Code Support) ist ein BS2000-Softwareprodukt für die Unterstützung internationaler Zeichensätze.

XML

XML (eXtensible Markup Language) ist eine vom W3C (WWW-Konsortium) genormte Metasprache, in der Austauschformate für Daten und zugehörige Informationen definiert werden können.

Zeitgesteuerter Auftrag
time-driven job

Auftrag, der von openUTM bis zu einem definierten Zeitpunkt in einer Message Queue zwischengespeichert und dann an den Empfänger weitergeleitet wird. Empfänger kann sein: ein Asynchron-Vorgang der selben Anwendung, eine TAC-Queue, eine Partner-Anwendung, ein Terminal oder ein Drucker. Zeitgesteuerte Aufträge können nur von KDCS-Teilprogrammen erteilt werden.

Zugangskontrolle
system access control

Prüfung durch openUTM, ob eine bestimmte Benutzerkennung berechtigt ist, mit der UTM-Anwendung zu arbeiten. Die Berechtigungsprüfung entfällt, wenn die UTM-Anwendung ohne Benutzerkennungen generiert wurde.

Zugriffskontrolle
data access control

Prüfung durch openUTM, ob der Kommunikationspartner berechtigt ist, auf ein bestimmtes Objekt der Anwendung zuzugreifen. Die Zugriffsrechte werden als Bestandteil der Konfiguration festgelegt.

Zugriffspunkt
access point

siehe Dienstzugriffspunkt.

17 Abkürzungen

ACSE	Association Control Service Element
AEQ	Application Entity Qualifier
AES	Advanced Encryption Standard
AET	Application Entity Title
APT	Application Process Title
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
Axis	Apache eXtensible Interaction System
BCAM	Basic Communication Access Method
BER	Basic Encoding Rules
BLS	Binder-Lader-Starter (BS2000-Systeme)
CCP	Communication Control Program
CCR	Commitment, Concurrency and Recovery
CCS	Codierter Zeichensatz (Coded Character Set)
CCSN	Name des codierten Zeichensatzes (Coded Character Set Name)
CICS	Customer Information Control System (IBM)
CID	Control Identification
CMX	Communication Manager in Unix-, Linux- und Windows-Systemen
COM	Component Object Model
CPI-C	Common Programming Interface for Communication
CRM	Communication Resource Manager
CRTE	Common Runtime Environment (BS2000-Systeme)
DB	Database
DBH	Database Handler
DC	Data Communication
DCAM	Data Communication Access Method
DES	Data Encryption Standard

DLS	Distributed Lock Manager (BS2000-Systeme)
DMS	Data Management System
DNS	Domain Name Service
DSS	Datensichtstation (=Terminal)
DTD	Document Type Definition
DTP	Distributed Transaction Processing
DVS	Datenverwaltungssystem
EBCDIC	Extended Binary-Coded Decimal Interchange Code
EJB	Enterprise JavaBeans™
FGG	File Generation Group
FHS	Format Handling System
FT	File Transfer
GCM	Galois/Counter Mode
GSSB	Globaler Sekundärer Speicherbereich
HIPLEX®	Highly Integrated System Complex (BS2000-Systeme)
HLL	High-Level Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IFG	Interaktiver Format-Generator
ILCS	Inter Language Communication Services (BS2000-Systeme)
IMS	Information Management System (IBM)
IPC	Inter-Process-Communication
IRV	Internationale Referenzversion
ISO	International Organization for Standardization
Java EE	Java Platform, Enterprise Edition
JCA	Java EE Connector Architecture
JDK	Java Development Kit
KAA	KDCS Application Area
KB	Kommunikationsbereich

KBPROG	KB-Programmbereich
KDCADMI	KDC Administration Interface
KDCS	Kompatible Datenkommunikationsschnittstelle
KTA	KDCS Task Area
LAN	Local Area Network
LCF	Local Configuration File
LLM	Link and Load Module (BS2000-Systeme)
LSSB	Lokaler Sekundärer Speicherbereich
LU	Logical Unit
MQ	Message Queuing
MSCF	Multiple System Control Facility (BS2000-Systeme)
NB	Nachrichtenbereich
NEA	Netzwerkarchitektur bei BS2000-Systemen
NFS	Network File System/Service
NLS	Unterstützung der Landessprache (Native Language Support)
OLTP	Online Transaction Processing
OML	Object Modul Library
OSI	Open System Interconnection
OSI TP	Open System Interconnection Transaction Processing
OSS	OSI Session Service
PCMX	Portable Communication Manager
PID	Prozess-Identifikation
PIN	Persönliche Identifikationsnummer
PLU	Primary Logical Unit
PTC	Prepare to commit
RAV	Rechenzentrums-Abrechnungs-Verfahren
RDF	Resource Definition File
RM	Resource Manager
RSA	Encryption-Algorithmus nach Rivest, Shamir, Adleman

RSO	Remote SPOOL Output (BS2000-Systeme)
RTS	Runtime System (Laufzeitsystem)
SAT	Security Audit Trail (BS2000-Systeme)
SECOS	Security Control System
SEM	SE Manager
SGML	Standard Generalized Markup Language
SLU	Secondary Logical Unit
SM2	Software Monitor 2
SNA	Systems Network Architecture
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
SPAB	Standard Primärer Arbeitsbereich
SQL	Structured Query Language
SSB	Sekundärer Speicherbereich
SSL	Secure Socket Layer
SSO	Single-Sign-On
TAC	Transaktionscode
TCEP	Transport Connection End Point
TCP/IP	Transport Control Protocol / Internet Protocol
TIAM	Terminal Interactive Access Method
TLS	Terminal-spezifischer Langzeitspeicher
TLS	Transport Layer Security
TM	Transaction Manager
TNS	Transport Name Service
TP	Transaction Processing (Transaktions-Betrieb)
TPR	Task privileged (privilegierter Funktionszustand des BS2000-Systems)
TPSU	Transaction Protocol Service User
TSAP	Transport Service Access Point
TSN	Task Sequence Number
TU	Task user (nicht privilegierter Funktionszustand des BS2000-Systems)

TX	Transaction Demarcation (X/Open)
UDDI	Universal Description, Discovery and Integration
UDS	Universelles Datenbanksystem
UDT	Unstructured Data Transfer
ULS	User-spezifischer Langzeitspeicher
UPIC	Universal Programming Interface for Communication
USP	UTM-Socket-Protokoll
UTM	Universeller Transaktionsmonitor
UTM-D	UTM-Funktionen für verteilte Verarbeitung („Distributed“)
UTM-F	Schnelle UTM-Variante („Fast“)
UTM-S	UTM-Sicherheitsvariante
UTM-XML	XML-Schnittstelle von openUTM
VGID	Vorgangs-Identifikation
VTSU	Virtual Terminal Support
VTV	Verteilte Transaktionsverarbeitung
VV	Verteilte Verarbeitung
WAN	Wide Area Network
WS4UTM	WebServices for openUTM
WSDD	Web Service Deployment Descriptor
WSDL	Web Services Description Language
XA	X/Open Access Interface (Schnittstelle von X/Open zum Zugriff auf Resource Manager)
XAP	X/OPEN ACSE/Presentation programming interface
XAP-TP	X/OPEN ACSE/Presentation programming interface Transaction Processing extension
XATMI	X/Open Application Transaction Manager Interface
XCS	Cross Coupled System
XHCS	eXtended Host Code Support
XML	eXtensible Markup Language

18 Literatur

Die Handbücher finden Sie im Internet unter https://bs2manuals.ts.fujitsu.com.

Dokumentation zu openUTM

openUTM
Konzepte und Funktionen
Benutzerhandbuch

openUTM
Anwendungen programmieren mit KDCS für COBOL, C und C++
Basishandbuch

openUTM
Anwendungen generieren
Benutzerhandbuch

openUTM
Einsatz von UTM-Anwendungen auf BS2000-Systemen
Benutzerhandbuch

openUTM
Einsatz von UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen
Benutzerhandbuch

openUTM
Anwendungen administrieren
Benutzerhandbuch

openUTM
Meldungen, Test und Diagnose auf BS2000-Systemen
Benutzerhandbuch

openUTM
Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen
Benutzerhandbuch

openUTM
Anwendungen erstellen mit X/Open-Schnittstellen
Benutzerhandbuch

openUTM
XML für openUTM

openUTM-Client (Unix-Systeme) **für Trägersystem OpenCPIC**
Client-Server-Kommunikation mit openUTM
Benutzerhandbuch

openUTM-Client für Trägersystem UPIC
Client-Server-Kommunikation mit openUTM
Benutzerhandbuch

openUTM WinAdmin
Grafischer Administrationsarbeitsplatz für openUTM
Beschreibung und Online-Hilfe

openUTM WebAdmin
Web-Oberfläche zur Administration von openUTM
Beschreibung und Online-Hilfe

openUTM, openUTM-LU62
Verteilte Transaktionsverarbeitung
zwischen openUTM und CICS-, IMS- und LU6.2-Anwendungen
Benutzerhandbuch

openUTM (BS2000)
Anwendungen programmieren mit KDCS für Assembler
Ergänzung zum Basishandbuch

openUTM (BS2000)
Anwendungen programmieren mit KDCS für Fortran
Ergänzung zum Basishandbuch

openUTM (BS2000)
Anwendungen programmieren mit KDCS für Pascal-XT
Ergänzung zum Basishandbuch

openUTM (BS2000)
Anwendungen programmieren mit KDCS für PL/I
Ergänzung zum Basishandbuch

WS4UTM (Unix- und Windows-Systeme**)**
Web-Services für openUTM

Dokumentation zum openSEAS-Produktumfeld

BeanConnect
Benutzerhandbuch

openUTM-JConnect
Verbindung von Java-Clients zu openUTM
Benutzerdokumentation und Java-Docs

WebTransactions
Konzepte und Funktionen

WebTransactions
Template-Sprache

WebTransactions
Anschluss an openUTM-Anwendungen über UPIC

WebTransactions
Anschluss an MVS-Anwendungen

WebTransactions
Anschluss an OSD-Anwendungen

Dokumentation zum BS2000-Umfeld

AID Advanced Interactive Debugger
Basishandbuch
Benutzerhandbuch

AID Advanced Interactive Debugger
Testen von COBOL-Programmen
Benutzerhandbuch

AID Advanced Interactive Debugger
Testen von C/C++-Programmen
Benutzerhandbuch

BCAM
BCAM Band 1/2
Benutzerhandbuch

BINDER
Benutzerhandbuch

BS2000 OSD/BC
Kommandos Band 1-7
Benutzerhandbuch

BS2000 OSD/BC
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

BS2IDE
Eclipse-based Integrated Development Environment for BS2000
User Guide and Installation Guide
Webseite: *https://bs2000.ts.fujitsu.com/bs2ide/*

BLSSERV
Bindelader-Starter in BS2000/OSD
Benutzerhandbuch

DCAM
COBOL-Aufrufe
Benutzerhandbuch

DCAM
Makroaufrufe
Benutzerhandbuch

DCAM
Programmschnittstellen
Beschreibung

FHS
Formatierungssystem für openUTM, TIAM, DCAM
Benutzerhandbuch

IFG für FHS
Benutzerhandbuch

HIPLEX AF
Hochverfügbarkeit von Anwendungen in BS2000/OSD
Produkthandbuch

HIPLEX MSCF
BS2000-Rechner im Verbund
Benutzerhandbuch

IMON
Installationsmonitor
Benutzerhandbuch

LMS
SDF-Format
Benutzerhandbuch

MT9750 (MS Windows)
9750-Emulation unter Windows
Produkthandbuch

OMNIS/OMNIS-MENU
Funktionen und Kommandos
Benutzerhandbuch

OMNIS/OMNIS-MENU
Administration und Programmierung
Benutzerhandbuch

OSS (BS2000)
OSI Session Service
User Guide

openSM2
Software Monitor
Benutzerhandbuch

RSO
Remote SPOOL Output
Benutzerhandbuch

SECOS
Security Control System
Benutzerhandbuch

SECOS
Security Control System
Tabellenheft

SESAM/SQL
Datenbankbetrieb
Benutzerhandbuch

TIAM
Benutzerhandbuch

UDS/SQL
Datenbankbetrieb
Benutzerhandbuch

Unicode im BS2000/OSD
Übersichtshandbuch

VTSU
Virtual Terminal Support
Benutzerhandbuch

XHCS
8-bit-Code- und Unicode-Unterstützung im BS2000/OSD
Benutzerhandbuch

Dokumentation zum Umfeld von Unix-, Linux- und Windows-Systemen

CMX V6.0 (Unix-Systeme)
Betrieb und Administration
Benutzerhandbuch

CMX V6.0
CMX-Anwendungen programmieren
Programmierhandbuch

OSS (UNIX)
OSI Session Service
User Guide

PRIMECLUSTER™
Konzept (Solaris, Linux)
Benutzerhandbuch

openSM2
Die Dokumentation zu openSM2 wird in Form von ausführlichen Online-Hilfen bereitgestellt, die mit dem Produkt ausgeliefert werden.

Sonstige Literatur

CPI-C
X/Open CAE Specification
Distributed Transaction Processing:
The CPI-C Specification, Version 2
ISBN 1 85912 135 7

Reference Model
X/Open Guide
Distributed Transaction Processing:
Reference Model, Version 2
ISBN 1 85912 019 9

REST
Architectural Styles and the Design of Network-based Software Architectures
Dissertation Roy Fielding

TX
X/Open CAE Specification
Distributed Transaction Processing:
The TX (Transaction Demarcation) Specification
ISBN 1 85912 094 6

XATMI
X/Open CAE Secification
Distributed Transaction Processing
The XATMI Specification
ISBN 1 85912 130 6

XML
Spezifikation des W3C (www – Konsortium)
Webseite: *http://www.w3.org/XML*